

Лабораторная работа 1. Установка и настройка распределенной системы. Простейшие операции и знакомство с функциональностью системы.

Цель: ознакомление с процессом установки и настройки распределенных систем, таких как Apache(Arenadata) Hadoop или Apache Spark. Изучить основные операции и функциональные возможности системы, что позволит понять принципы работы с данными и распределенными вычислениями.

Необходимое ПО:

- Ubuntu 24.04 LTS (22.04, 20.04) или новее.
- Java 8 ил Java11 или новее.
- Apache Spark 3.4.3.
- Python 3.12+.
- pip (менеджер пакетов Python).

Алгоритм выполнения задания в Apache Spark

1. Установка Java
2. Установка Python и pip
3. Установка Apache Spark 3.4.3
4. Настройка переменных окружения
5. Загрузка экономических данных
6. Запуск Spark и выполнение простейших операций

1. Установка Java.

```
```bash
sudo apt update
sudo apt install openjdk-11-jdk
java -version
```
```

2. Установка Python и pip.

```
```bash
sudo apt install python3 python3-pip
python3 --version
pip3 --version
```
```

3. Установка Apache Spark 3.4.3:

```
```bash
wget https://downloads.apache.org/spark/spark-3.4.3/spark-3.4.3-bin-hadoop3.tgz
tar xvf spark-3.4.3-bin-hadoop3.tgz
sudo mv spark-3.4.3-bin-hadoop3 /opt/spark
```
```

4. Настройка переменных окружения:

Откройте файл ~/.bashrc:

```
```bash
nano ~/.bashrc
```
```

Добавьте следующие строки в конец файла:

```
```bash
export SPARK_HOME=/opt/spark
export PATH=$PATH:$SPARK_HOME/bin
```
```

...

...

...

///

...

2

b. Подсчет количества строк.

```
```python
print("Количество строк:", df.count())
```
```

c. Вывод схемы данных.

```
```python
df.printSchema()
```
```

```
>>> print("Количество строк:", df.count())
Количество строк: 11016
>>> df.printSchema()
root
 |-- Date: date (nullable = true)
 |-- Open: double (nullable = true)
 |-- High: double (nullable = true)
 |-- Low: double (nullable = true)
 |-- Close: double (nullable = true)
 |-- Adj Close: double (nullable = true)
 |-- Volume: long (nullable = true)
```

d. Базовая статистика.

```
```python
df.describe().show()
```
```

```
>>> df.describe().show()
24/08/26 12:00:52 WARN package: Truncated the string representation of a plan since it was too large. This behavior can be adjusted
+-----+-----+-----+-----+-----+-----+-----+
|summary|      Open|      High|      Low|      Close|      Adj Close|      Volume|
+-----+-----+-----+-----+-----+-----+-----+
|count|      11016|      11016|      11016|      11016|      11016|      11016|
|mean| 22.620120899782137| 22.865000826706627| 22.38652977641613| 22.635557646241836| 21.818456401960802| 3.176483384259259E8|
|stddev| 46.64693297111878| 47.152105248209864| 46.17963332460106| 46.687681156403094| 46.22436567146756| 3.3534976767000806E8|
|min|      0.049665|      0.049665|      0.049107|      0.049107|      0.037856|      0|
|max|      236.479996|      237.229996|      233.089996|      234.820007|      234.548523|      7421640800|
+-----+-----+-----+-----+-----+-----+-----+
```

e. Фильтрация данных.

```
```python
df_filtered = df.filter(df["Date"] >= "2020-01-01")
df_filtered.show(5)
```
```

```
>>> df_filtered = df.filter(df["Date"] >= "2020-01-01")
>>> f["Date"] >= "2020-01-01"
File "<stdin>", line 1
  f["Date"] >= "2020-01-01"
                        ^
SyntaxError: unmatched ')
>>> df_filtered.show(5)
+-----+-----+-----+-----+-----+-----+-----+
|Date|Open|High|Low|Close|Adj Close|Volume|
+-----+-----+-----+-----+-----+-----+-----+
|2020-01-02|74.059998|75.150002|73.797501|75.087502|72.876114|135480400|
|2020-01-03|74.287498|75.144997|74.125|74.357498|72.16761|146322800|
|2020-01-06|73.447502|74.989998|73.1875|74.949997|72.742653|118387200|
|2020-01-07|74.959999|75.224998|74.370003|74.597504|72.400536|108872000|
|2020-01-08|74.290001|76.110001|74.290001|75.797501|73.565178|132079200|
+-----+-----+-----+-----+-----+-----+-----+
```

f. Группировка и агрегация.

```
```python
from pyspark.sql.functions import year, avg
df_yearly = df.withColumn("Year",
year(df["Date"])).groupBy("Year").agg(avg("Close").alias("Avg_Close"))
df_yearly.orderBy("Year").show()
```
```

| Year | Avg_Close |
|------|---------------------|
| 1980 | 0.13590307692307693 |
| 1981 | 0.10854781818181822 |
| 1982 | 0.08545888142292486 |
| 1983 | 0.1672740118577075 |
| 1984 | 0.1196512490118577 |
| 1985 | 0.09023348809523808 |
| 1986 | 0.14491285375494065 |
| 1987 | 0.3477511106719368 |
| 1988 | 0.3708842411067196 |
| 1989 | 0.3719529682539681 |
| 1990 | 0.33537290513834017 |
| 1991 | 0.46870147826086955 |
| 1992 | 0.4893094763779526 |
| 1993 | 0.36630964426877505 |
| 1994 | 0.3042905634920634 |
| 1995 | 0.36196595238095214 |
| 1996 | 0.22249534645669278 |
| 1997 | 0.1604176679841897 |
| 1998 | 0.2729004801587301 |
| 1999 | 0.515805492063492 |

g. Создание временного представления и выполнение SQL-запроса:

```
```python
df.createOrReplaceTempView("stock_data")
spark.sql("SELECT Year(Date) as Year, AVG(Close) as Avg_Close FROM stock_data GROUP BY
Year(Date) ORDER BY Year").show()
```
```

| Year | Avg_Close |
|------|---------------------|
| 1980 | 0.13590307692307693 |
| 1981 | 0.10854781818181822 |
| 1982 | 0.08545888142292486 |
| 1983 | 0.1672740118577075 |
| 1984 | 0.1196512490118577 |
| 1985 | 0.09023348809523808 |
| 1986 | 0.14491285375494065 |
| 1987 | 0.3477511106719368 |
| 1988 | 0.3708842411067196 |
| 1989 | 0.3719529682539681 |
| 1990 | 0.33537290513834017 |
| 1991 | 0.46870147826086955 |
| 1992 | 0.4893094763779526 |
| 1993 | 0.36630964426877505 |
| 1994 | 0.3042905634920634 |
| 1995 | 0.36196595238095214 |
| 1996 | 0.22249534645669278 |
| 1997 | 0.1604176679841897 |
| 1998 | 0.2729004801587301 |
| 1999 | 0.515805492063492 |

Чтобы выйти из среды PySpark, выполните следующие действия:

1. Если вы работаете в интерактивной сессии PySpark (например, через терминал или Jupyter Notebook), просто введите:

```
```python
spark.stop()
```
```

Это завершит сеанс Spark.

2. После этого можете выйти из Python интерактивной оболочки:

```
```python
exit()
```
```

Или просто нажмите `Ctrl + D`. Это завершит сессию PySpark и закроет текущий Python интерпретатор.

Алгоритм выполнения задания в Apache Hadoop

В виртуальной машине Шаг 1-8 пропустить.

Шаг 1. Установка необходимых компонентов.

```
```bash
```

```
sudo apt update
sudo apt install ssh pdsh -y
```
```

Шаг 2. Создание пользователя Hadoop.

```
```bash
sudo adduser hadoop
sudo usermod -aG sudo hadoop
su - hadoop
```
```

Шаг 3. Настройка SSH.

```
```bash
ssh-keygen -t rsa -P ""
cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
chmod 0600 ~/.ssh/authorized_keys
```
```

Шаг 4. Загрузка и установка Hadoop.

```
```bash
wget https://downloads.apache.org/hadoop/common/hadoop-3.3.5/hadoop-3.3.5.tar.gz
tar -xvzf hadoop-3.3.5.tar.gz
sudo mv hadoop-3.3.5 /usr/local/hadoop
```
```

Шаг 5. Настройка окружения Hadoop.

Добавьте следующие строки в конец файла ~/.bashrc:

```
```bash
export HADOOP_HOME=/usr/local/hadoop
export HADOOP_INSTALL=$HADOOP_HOME
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin
```
```

Применить изменения:

```
```bash
source ~/.bashrc
```
```

Шаг 6. Настройка конфигурационных файлов Hadoop.

a) Отредактируйте \$HADOOP_HOME/etc/hadoop/hadoop-env.sh:

```
```bash
export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64
```
```

b) Отредактируйте \$HADOOP_HOME/etc/hadoop/core-site.xml:

```
```xml
```

```
<configuration>
 <property>
 <name>fs.defaultFS</name>
 <value>hdfs://localhost:9000</value>
 </property>
</configuration>
...

```

c) Отредактируйте \$HADOOP\_HOME/etc/hadoop/hdfs-site.xml:

```
```xml
<configuration>
  <property>
    <name>dfs.replication</name>
    <value>1</value>
  </property>
  <property>
    <name>dfs.namenode.name.dir</name>
    <value>/home/hadoop/hdfs/namenode</value>
  </property>
  <property>
    <name>dfs.datanode.data.dir</name>
    <value>/home/hadoop/hdfs/datanode</value>
  </property>
</configuration>
...

```

d) Отредактируйте \$HADOOP_HOME/etc/hadoop/mapred-site.xml:

```
```xml
<configuration>
 <property>
 <name>mapreduce.framework.name</name>
 <value>yarn</value>
 </property>
</configuration>
...

```

e) Отредактируйте \$HADOOP\_HOME/etc/hadoop/yarn-site.xml:

```
```xml
<configuration>
  <property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
  </property>
</configuration>
...

```

Шаг 7. Создание директорий для HDFS.

```
```bash
mkdir -p ~/hdfs/namenode ~/hdfs/datanode
...

```

Шаг 8. Форматирование HDFS.

```
```bash
hdfs namenode -format
```
```

Все дальнейшие действия выполняются пользователем **hadoop**.

```
```bash
sudo su - hadoop
```
```

Шаг 9. Запуск Hadoop.

```
```bash
start-dfs.sh
start-yarn.sh
```
```

Шаг 10. Проверка работы Hadoop.

```
```bash
jps
```
```

```
hadoop@devopsvm:~$ jps
5682 SecondaryNameNode
5940 ResourceManager
6068 NodeManager
11236 SparkSubmit
5465 DataNode
9145 SparkSubmit
10250 SparkSubmit
5181 NameNode
12478 Jps
9422 SparkSubmit
```

Вы должны увидеть следующие процессы: NameNode, DataNode, SecondaryNameNode, ResourceManager, NodeManager.

В стандартной конфигурации Hadoop HDFS предоставляет веб-интерфейс, доступный через веб-браузер на порту 9870. Этот интерфейс позволяет просматривать состояние и структуру HDFS, а также выполнять некоторые операции.

Чтобы получить доступ к веб-интерфейсу HDFS, выполните следующие шаги:

- Убедитесь, что Hadoop (в частности, HDFS) запущен.
- Откройте веб-браузер на компьютере, с которого у вас есть сетевой доступ к серверу Hadoop.
- В адресной строке браузера введите:

```

<http://localhost:9870>

```

[localhost:9870/dfshealth.html#tab-overview](http://localhost:9870/dfshealth.html#tab-overview)



## Overview 'localhost:9000' (✓active)

|                |                                                            |
|----------------|------------------------------------------------------------|
| Started:       | Mon Aug 26 13:07:51 +0300 2024                             |
| Version:       | 3.3.5, r706d88266abcee09ed78fbaa0ad5f74d818ab0e9           |
| Compiled:      | Wed Mar 15 18:56:00 +0300 2023 by stevel from branch-3.3.5 |
| Cluster ID:    | CID-60a52b68-6139-4947-8731-3c039547a32e                   |
| Block Pool ID: | BP-1830111676-127.0.1.1-1724666841903                      |

- Если обращаетесь к Hadoop с другого компьютера, замените "localhost" на IP-адрес или имя хоста сервера, на котором запущен Hadoop.
- Нажмите Enter, должны увидеть веб-интерфейс HDFS.

Через этот веб-интерфейс вы сможете просматривать структуру директорий HDFS, проверять состояние и здоровье узлов, просматривать логи и выполнять другие административные задачи.

Шаг 11. Работа с экономическими данными

а) Создайте директорию в HDFS:

```
```bash
hdfs dfs -mkdir /user
hdfs dfs -mkdir /user/hadoop
hdfs dfs -mkdir /user/hadoop/input
```
```

## Browse Directory

The screenshot shows the HDFS web interface for the directory /user/hadoop/input. At the top, there is a search bar with the text "/user/hadoop/input" and a "Go!" button. Below the search bar, there are icons for file operations. A table header is visible with columns: Permission, Owner, Group, Size, Last Modified, Replication, Block Size, and Name. The table body is empty, with the message "No data available in table" and "Showing 0 to 0 of 0 entries". At the bottom right, there are "Previous" and "Next" buttons. The footer of the interface says "Hadoop, 2023."

Проведем расчет экономических показателей на примере открытых экономических данных. Будем использовать данные о ВВП стран мира от Всемирного банка.

Шаг 12. Подготовка данных.

Скачайте данные о ВВП стран мира

([https://raw.githubusercontent.com/BosenkoTM/Distributed\\_systems/main/practice/2024/lw\\_01/GDP.csv](https://raw.githubusercontent.com/BosenkoTM/Distributed_systems/main/practice/2024/lw_01/GDP.csv)). Сохраните файл как **GDP.csv**.

Шаг 13. Загрузка данных в HDFS.

```
```bash
wget
https://raw.githubusercontent.com/BosenkoTM/Distributed_systems/main/practice/2024/lw_01/GDP.csv
hdfs dfs -mkdir /user/hadoop/economic_data
hdfs dfs -put GDP.csv /user/hadoop/economic_data/
```
```

Обработка данных с помощью MapReduce или Spark

Шаг 14. Запустите Spark.

```
```bash
spark-shell
```
```

Шаг 15. Загрузите данные и выполните расчеты.

```
```scala
val data = spark.read.option("header", "true").csv("file:///home/hadoop/GDP.csv")
Вычисляем среднее значение GDR
val result = data.selectExpr("avg(GDR) as avg_GDR")
```
```



```
scala> val data = spark.read.option("header","true").csv("file:///home/hadoop/GDP.csv")
data: org.apache.spark.sql.DataFrame = [Country: string, Year: string ... 21 more fields]

scala> val result = data.selectExpr("avg(GDP) as avg_GDR")
result: org.apache.spark.sql.DataFrame = [avg_GDR: double]
```

```
// Сохраняем результат в CSV файл
result.write.option("header", "true").csv("/home/hadoop/output/avg_GDR.csv")
...

```

Выходим из Scala.

```
```scala
:q
...

```

В Ubuntu 24.04 Scala сохраняет результаты в файле **part-00000-*.csv**, каталог будет определен последним адресом в пути при сохранении, то есть **avg_GDR.csv**.

Шаг 16. Переименовать полученный результат **part-00000-*.csv** в Ubuntu **avg.csv**

```
```bash
mv part-00000-*.csv avg.csv
...

```

Шаг 17. **Переносим данные в HDFS.** Загрузите экономические данные в HDFS:

```
```bash
hdfs dfs -put /home/hadoop/output/avg.csv /user/hadoop/input/
...

```

Проверьте, что данные загружены.

```
```bash
hdfs dfs -ls /user/hadoop/input/
...

```

/user/hadoop/input

Go!

Show

25

entries

Search:

<input type="checkbox"/>	<div><div></div><div></div><div></div></div> Permission	<div><div></div><div></div><div></div></div> Owner	<div><div></div><div></div><div></div></div> Group	<div><div></div><div></div><div></div></div> Size	<div><div></div><div></div><div></div></div> Last Modified	<div><div></div><div></div><div></div></div> Replication	<div><div></div><div></div><div></div></div> Block Size	<div><div></div><div></div><div></div></div> Name	<div><div></div><div></div><div></div></div>
<input type="checkbox"/>	-rw-r--r--	hadoop	supergroup	27 B	Aug 26 15:42	1	128 MB	avg.csv	<div><div></div><div></div><div></div></div>

Чтобы остановить Hadoop 3 в Ubuntu, выполните следующие шаги.

Сначала остановите YARN (если он запущен).

```
```bash
stop-yarn.sh

```

Затем остановите HDFS.

```
```bash
stop-dfs.sh

```

Если вы используете MapReduce JobHistory Server, остановите его:

```
mr-jobhistory-daemon.sh stop historyserver.
```

Для полной остановки всех Hadoop-демонов можно использовать команду.

```
```bash
stop-all.sh

```

Проверьте, что все процессы Hadoop остановлены.

```
```bash
jps

```

Эта команда покажет список запущенных Java-процессов. Убедитесь, что в списке нет процессов, связанных с Hadoop (например, NameNode, DataNode, ResourceManager и т.д.).

Если какие-то процессы остались, вы можете остановить их вручную с помощью команды kill.

```
```bash
```

```
kill -9 <PID>
```

где <PID> - идентификатор процесса, который вы хотите остановить.

Задание для самостоятельной работы

1. Загрузите данные по акциям другой компании (например, Microsoft - MSFT).
2. Выполните аналогичный анализ для новых данных.
3. Сравните результаты анализа двух компаний.
4. Напишите Spark-приложение, которое находит дни с максимальным объемом торгов для обеих компаний.

Отчет

Студенты должны подготовить отчет, включающий:

1. Описание процесса установки и настройки Spark или Hadoop.
2. Листинг выполненных команд и их результаты.
3. Анализ полученных результатов.
4. Код и результаты выполнения задания для самостоятельной работы.
5. Выводы о функциональности и возможностях Apache Spark или Hadoop для анализа экономических данных.

Варианты заданий

Вариант выбирается согласно номеру студента в списке группы:

1. Установка Apache Spark на одном узле и выполнение простой задачи на подсчет строк в файле.

Данные: Исторические данные по акциям Сбербанка (SBER) с сайта Московской биржи (moex.com)

Операции: Фильтрация данных за 2020 год, расчет средней цены закрытия, группировка по месяцам.

2. Установка Apache(Arenadata) Hadoop и выполнение задачи на копирование файлов в HDFS.

Данные: Исторические данные по акциям Газпрома (GAZP) с сайта Московской биржи (moex.com)

Операции: Фильтрация данных за 2019 год, расчет максимальной цены открытия, группировка по кварталам.

3. Установка Apache Spark и выполнение задачи на сортировку данных.

Данные: Исторические данные по акциям Лукойла (LKOH) с сайта Московской биржи (moex.com)

Операции: Фильтрация данных за последние 5 лет, расчет минимальной цены закрытия, группировка по годам.

4. Настройка кластерного режима для Apache Spark на 2 узлах.

Данные: Исторические данные по акциям Яндекса (YNDX) с сайта Московской биржи (moex.com)

Операции: Фильтрация данных за 2021 год, расчет средней цены закрытия, тренд анализа.

5. Настройка кластерного режима для Apache(Arenadata) Hadoop на 2 узлах и проверка работоспособности.
Данные: Исторические данные по акциям Роснефти (ROSN) с сайта Московской биржи (moex.com)
Операции: Фильтрация данных за последние 3 года, расчет медианной цены закрытия, группировка по месяцам.
6. Установка Apache Spark на кластере из 3 узлов и выполнение задачи на агрегацию данных.
Данные: Исторические данные по акциям Норильского никеля (GMKN) с сайта Московской биржи (moex.com)
Операции: Фильтрация данных за 2020 год, расчет стандартного отклонения цены закрытия, группировка по кварталам.
7. Установка и настройка Apache Spark для работы с внешним источником данных (например, S3).
Данные: Исторические данные по акциям ВТБ (VTBR) с сайта Московской биржи (moex.com)
Операции: Фильтрация данных за последние 10 лет, расчет коэффициента вариации цены закрытия, тренд
8. Установка Apache(Arenadata) Hadoop и выполнение задачи на объединение файлов в HDFS.
Данные: Исторические данные по акциям Магнита (MGNT) с сайта Московской биржи (moex.com)
Операции: Фильтрация данных за 2018 год, расчет средней цены открытия, группировка по годам.
9. Установка и настройка Apache Spark для работы с Cassandra.
Данные: Исторические данные по акциям Полюса (PLZL) с сайта Московской биржи (moex.com)
Операции: Фильтрация данных за 2019 год, расчет средней цены закрытия, корреляция с объемом торгов.
10. Настройка Apache Spark для работы с SQL-запросами.
Данные: Исторические данные по акциям МТС (MTSS) с сайта Московской биржи (moex.com)
Операции: Фильтрация данных за последние 2 года, расчет максимальной цены закрытия, тренд анализа.
11. Установка Apache(Arenadata) Hadoop и выполнение задачи на создание и удаление каталогов в HDFS.
Данные: Исторические данные по акциям Татнефти (TATN) с сайта Московской биржи (moex.com)
Операции: Фильтрация данных за последние 5 лет, расчет минимальной цены закрытия, группировка по месяцам.
12. Установка Apache Spark и выполнение задачи на чтение и запись данных из/в Parquet.
Данные: Исторические данные по акциям Сургутнефтегаз (SNGS) с сайта Московской биржи (moex.com)
Операции: Фильтрация данных за 2020 год, расчет стандартного отклонения цены открытия, тренд анализа.
13. Настройка кластерного режима для Apache Spark на 4 узлах с разными ролями узлов.
Данные: Исторические данные по акциям Мечела (MTLR) с сайта Московской биржи (moex.com)
Операции: Фильтрация данных за последние 3 года, расчет медианной цены закрытия, группировка по кварталам.
14. Установка Apache(Arenadata) Hadoop и выполнение задачи на распределение файлов между узлами.

Данные: Исторические данные по акциям Интер PAO (IRAO) с сайта Московской биржи (moex.com)

Операции: Фильтрация данных за 2018 год, расчет средней цены открытия, корреляция с объемом торгов.

15. Установка Apache Spark и выполнение задачи на анализ текстовых данных.

Данные: Исторические данные по акциям Аэрофлота (AFLT) с сайта Московской биржи (moex.com)

Операции: Фильтрация данных за последние 2 года, расчет максимальной цены закрытия, тренд анализа.

16. Настройка кластерного режима для Apache Spark на 3 узлах с использованием Docker.

Данные: Исторические данные по акциям Системы (AFKS) с сайта Московской биржи (moex.com)

Операции: Фильтрация данных за 2019 год, расчет средней цены закрытия, группировка по месяцам.

17. Установка Apache(Arenadata) Hadoop и выполнение задачи на создание и просмотр логов системы.

Данные: Исторические данные по акциям ФосАгро (PHOR) с сайта Московской биржи (moex.com)

Операции: Фильтрация данных за последние 5 лет, расчет минимальной цены закрытия, тренд анализа.

18. Установка Apache Spark и выполнение задачи на фильтрацию данных.

Данные: Исторические данные по акциям Алросы (ALRS) с сайта Московской биржи (moex.com)

Операции: Фильтрация данных за 2020 год, расчет стандартного отклонения цены закрытия, группировка

19. Настройка кластерного режима для Apache(Arenadata) Hadoop на 4 узлах и выполнение задачи на распределенную обработку данных.

Данные: Исторические данные по акциям Русала (RUAL) с сайта Московской биржи (moex.com)

Операции: Фильтрация данных за последние 3 года, расчет медианной цены закрытия, корреляция с объемом торгов

20. Установка Apache Spark и выполнение задачи на работу с JSON-файлами.

Данные: Исторические данные по акциям Мосбиржи (MOEX) с сайта Московской биржи (moex.com)

Операции: Фильтрация данных за 2018 год, расчет средней цены открытия, группировка по годам.

21. Настройка Apache Spark для работы с Hive.

Данные: Исторические данные по акциям РУСГИДРО (HYDR) с сайта Московской биржи (moex.com)

Операции: Фильтрация данных за последние 10 лет, расчет коэффициента вариации цены закрытия, тренд анализа.

22. Установка Apache(Arenadata) Hadoop и выполнение задачи на создание резервной копии данных.

Данные: Исторические данные по акциям Россетей (RSTI) с сайта Московской биржи (moex.com)

Операции: Фильтрация данных за 2021 год, расчет максимальной цены закрытия, корреляция с объемом торгов.

23. Установка Apache Spark и выполнение задачи на вычисление статистических параметров данных.

Данные: Исторические данные по акциям X5 Retail Group (FIVE) с сайта Московской биржи (moex.com)

Операции: Фильтрация данных за последние 2 года, расчет средней цены закрытия, группировка по месяцам.

24. Настройка кластерного режима для Apache Spark на 2 узлах с использованием Ansible.

Данные: Исторические данные по акциям ТМК (TRMK) с сайта Московской биржи (moex.com)

Операции: Фильтрация данных за 2019 год, расчет минимальной цены закрытия, тренд анализа.

25. Установка Apache(Arenadata) Hadoop и выполнение задачи на слияние данных из нескольких источников в HDFS.

Данные: Исторические данные по акциям М.Видео (MVID) с сайта Московской биржи (moex.com)

Операции: Фильтрация данных за последние 5 лет, расчет стандартного отклонения цены закрытия, группировка по кварталам.