

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 -xzf 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>
```
```

```
    </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>
```

```
...
```

е) 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
```

```
...
```

Шаг 9: Запуск Hadoop

```
```bash
```

```
start-dfs.sh
```

```
start-yarn.sh
```

```
...
```

Шаг 10: Проверка работы Hadoop

```
```bash
```

```
jps
```

```
```
```

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
```

```
```
```

- Если вы обращаетесь к Hadoop с другого компьютера, замените "localhost" на IP-адрес или имя хоста сервера, на котором запущен Hadoop.
- Нажмите Enter, и вы 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
```

...

Проведем расчет экономических показателей на примере открытых экономических данных с использованием Hadoop 3 в Ubuntu. Мы будем использовать данные о ВВП стран мира от Всемирного банка.

#### 1. Подготовка данных:

Скачайте данные о ВВП стран мира ([https://github.com/BosenkoTM/Distributed\\_systems/blob/main/practice/2024/Iw\\_01/GDP.csv](https://github.com/BosenkoTM/Distributed_systems/blob/main/practice/2024/Iw_01/GDP.csv)). Сохраните файл как **GDP.csv**.

#### 2. Загрузка данных в HDFS:

```
```bash
wget
https://github.com/BosenkoTM/Distributed_systems/blob/main/practice/2024/Iw_01/GDP.csv
```

```
hdfs dfs -mkdir /user/hadoop/economic_data
hdfs dfs -put GDP.csv /user/hadoop/economic_data/
```

...

Скачайте данные в формате CSV.

Загрузка данных в HDFS. После скачивания данных загрузите их в HDFS:

```
```bash
hdfs dfs -mkdir /user/hadoop/economic_data
hdfs dfs -put GDP.csv /user/hadoop/economic_data
```

...

Обработка данных с помощью MapReduce или Spark

#### 1. **\*\*Запустите Spark:\*\***

```
```bash
/usr/local/spark/bin/spark-shell
```

...

2. ****Загрузите данные и выполните расчеты:****

```
```scala
```

```
val data = spark.read.option("header",
"true").csv("/user/hadoop/economic_data/GDP.csv")

// Пример: расчет среднего значения по столбцу GDP
data.selectExpr("avg(GDP)").show()
```
```

Вывод результатов. После обработки данных, результаты можно сохранить или вывести.

```
```scala  
data.write.csv("/user/hadoop/economic_data/results")
```
```

Результаты сохранятся в HDFS и их можно будет загрузить на локальный компьютер для дальнейшего анализа.

b) Загрузите экономические данные в HDFS:

```
```bash  
hdfs dfs -put /path/to/economic_data.csv /user/hadoop/input/
```
```

c) Проверьте, что данные загружены:

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

Шаг 12: Выполнение простого MapReduce задания

Создайте простой Java-класс для подсчета слов в ваших экономических данных. Назовите файл WordCount.java:

```
```java
```

```
import java.io.IOException;
```

```
import java.util.StringTokenizer;
```

```
import org.apache.hadoop.conf.Configuration;
```

```
import org.apache.hadoop.fs.Path;
```

```
import org.apache.hadoop.io.IntWritable;
```

```
import org.apache.hadoop.io.Text;
```

```
import org.apache.hadoop.mapreduce.Job;
```

```
import org.apache.hadoop.mapreduce.Mapper;
```

```
import org.apache.hadoop.mapreduce.Reducer;
```

```
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
```

```
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
```

```
public class WordCount {
```

```
 public static class TokenizerMapper
```

```
 extends Mapper<Object, Text, Text, IntWritable>{
```

```
 private final static IntWritable one = new IntWritable(1);
```

```
 private Text word = new Text();
```

```
 public void map(Object key, Text value, Context context
```

```
) throws IOException, InterruptedException {
```

```
 StringTokenizer itr = new StringTokenizer(value.toString());
```

```
 while (itr.hasMoreTokens()) {
```

```
 word.set(itr.nextToken());
```

```
 context.write(word, one);
```



```
 }
 }
}
```

```
public static class IntSumReducer
 extends Reducer<Text,IntWritable,Text,IntWritable> {
 private IntWritable result = new IntWritable();

 public void reduce(Text key, Iterable<IntWritable> values,
 Context context
) throws IOException, InterruptedException {
 int sum = 0;
 for (IntWritable val : values) {
 sum += val.get();
 }
 result.set(sum);
 context.write(key, result);
 }
}
```

```
public static void main(String[] args) throws Exception {
 Configuration conf = new Configuration();
 Job job = Job.getInstance(conf, "word count");
 job.setJarByClass(WordCount.class);
 job.setMapperClass(TokenizerMapper.class);
 job.setCombinerClass(IntSumReducer.class);
 job.setReducerClass(IntSumReducer.class);
 job.setOutputKeyClass(Text.class);
 job.setOutputValueClass(IntWritable.class);
}
```

```
FileInputFormat.addInputPath(job, new Path(args[0]));
FileOutputFormat.setOutputPath(job, new Path(args[1]));
System.exit(job.waitForCompletion(true) ? 0 : 1);
}
}
...

```

Скомпилируйте и запустите задание:

```
```bash
hadoop com.sun.tools.javac.Main WordCount.java
jar cf wc.jar WordCount*.class
hadoop jar wc.jar WordCount /user/hadoop/input /user/hadoop/output
...

```

Шаг 13: Просмотр результатов

```
```bash
hdfs dfs -cat /user/hadoop/output/*
...

```

4. Задания для самостоятельной работы:

- Модифицируйте программу WordCount для подсчета частоты встречаемости определенных экономических терминов в ваших данных.
- Напишите MapReduce программу для расчета среднего значения выбранного экономического показателя.
- Создайте программу для поиска максимального и минимального значения экономического показателя в ваших данных.