First steps in Map-Reduce with Hadoop

LSDM 2025-2026

We suppose you use IntelliJ and Java.

1 Installing Hadoop

You are given several options for installing Hadoop. You need one working install, and not all of them.

1.1 Requirements

Hadoop requires Java 8 or Java 11.

These versions can be downloaded from at Oracle.

1.2 Native Installation (option 1)

1.2.1 Operating System

The default Hadoop environment is Linux.

It is possible to install on Mac OSX.

Using Windows is at your own risk as it presents multiple problems.

1.2.2 Linux Installation

The presented instructions follow the official site.

1. Download the current binary distribution of Hadoop from the official site
   `https://downloads.apache.org/hadoop/common/current/hadoop-X.Y.Z.tar.gz`
   or locally `https://filesender.renater.fr/?s=download&token=2daccdd5-6f7e-4180-8bd`

   Locally you will find the current version in 2025 i.e 3.4.2.

2. Extract the archive :

   ```
   tar zxvf hadoop-X.Y.Z.tar.gz
   ```

3. Configure your environment variables (in `$HOME/.bashrc` if you use `bash` or teh equivalent for other shells):

   ```
   export HADOOP_HOME=PATH_TO_DIR/hadoop-X.Y.Z
   export PATH=${HADOOP_HOME}/bin:$PATH
   ```

   `PATH_TO_DIR` is the directory where you have extracted Hadoop.

4. To apply the environment modifications

   ```
   source $HOME/.bashrc
   ```

5. Try the command

   ```
   hadoop
   ```

   to test your installation.

1.3 Installing Hadoop in a Docker container (option 2)

The major steps, detailed in the following, are:

1. Install Docker
2. Generate the Docker image
3. Test

1.3.1 Docker Installation

Docker is a technology that provides containers. Containers enable virtualization at a lower cost compared to virtual machines. Docker allows the execution of images that correspond to software that is already installed and configured.

Docker may be installed on all commun operating systems (Linux, Windows, MacOS). Follow the instructions given at `https://docs.docker.com/install`.

To install Docker on Ubuntu, go to
`https://docs.docker.com/install/linux/docker-ce/ubuntu/#install-docker-ce`

By default Docker containers can be launched only by *root*. To make this possible for a regular user, see
`https://docs.docker.com/install/linux/linux-postinstall/`

To test your Docker installation

```
docker run hello-world
```

To learn more on Docker : `https://docs.docker.com/engine/docker-overview/`

1.3.2 Génération de l'image Hadoop

ATTENTION The following instructions create a container in which it is possible to execute Hadoop jobs after they have been developed, compiled and packaged. If you wish to also be able to compile in the container, you will need to install the JDK (*Java Development Kit*) ein addition to the JRE (*Java Runtime Environment*).

Here are the steps:

1. Create a working directory for the docker image. Place in the directory the `Dockerfile` you can download at `https://filesender.renater.fr/?s=download&token=70b536ce-c953-40e`
   The `Dockerfile` is also given in appendix.

2. Execute

   ```
       docker build -t hadoop-container:1.0 .
   ```

   Be patient, the download of the components and the package installation take time.

3. Laucnh the container

   ```
   docker run -it hadoop-container:1.0 bash
   ```

To see the container's identifier :

```
docker ps
```

To copy data from you system to the container

```
docker cp LOCAL_PATH CONTAINER_ID:CONTAINER_PATH
```

1.3.3 Test the installation

In the container run

```
hadoop
```

2 Your first MapReduce Project

In the archive you have downloaded for teh Dockerfile, you will also find the `WordCount` official Apache example.

You also have the `pom.xml` for your `maven` project.

1. Compile.
   If you use the command line:

   ```
   mvn clean package
   ```

   Otherwise, in IntelliJ, use the graphical interface.

2. Create a directory for the input data (`input`) and place some text files.

   ```
   mkdir input
   ```

3. Execute. With the following command line, the created result directory wil be called `output`

   ```
   hadoop jar target/HadoopWordCount-1.0-SNAPSHOT.jar WordCount input output
   ```

ANNEXE : Dockerfile pour le conteneur Docker

```
FROM ubuntu:latest
LABEL maintainer vania.marangozova@univ-grenoble-alpes

RUN apt-get update --fix-missing &&\
apt-get install -y apt-utils &&\
apt-get install software-properties-common -y &&\
apt-get install -y openjdk-8-jre-headless ca-certificates-java &&\
apt-get install -y curl vim tar net-tools ssh &&\
apt clean &&\
    echo "Downloading and installing Hadoop..." && \
    curl -q https://downloads.apache.org/hadoop/core/hadoop-3.4.2/hadoop-3.4.2.tar.gz  |
tar xz -C /opt

RUN ln -s /opt/hadoop-3.4.2 /opt/hadoop

ARG PATH

ENV JAVA_HOME=/usr/lib/jvm/java-8-openjdk-arm64 \
    HDFS_HOME=/opt/hadoop \
    PATH=$PATH:/opt/hadoop/sbin:/opt/hadoop/bin
```

ANNEXE : WordCount.java

```java
/**
 * Licensed to the Apache Software Foundation (ASF) under one
 * or more contributor license agreements.  See the NOTICE file
 * distributed with this work for additional information
 * regarding copyright ownership.  The ASF licenses this file
 * to you under the Apache License, Version 2.0 (the
 * "License"); you may not use this file except in compliance
 * with the License.  You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */
//package org.apache.hadoop.examples;

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;
import org.apache.hadoop.util.GenericOptionsParser;

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();
        String[] otherArgs = new GenericOptionsParser(conf, args).getRemainingArgs();
        if (otherArgs.length < 2) {
            System.err.println("Usage: wordcount <in> [<in>...] <out>");
            System.exit(2);
        }
        Job job = Job.getInstance(conf, "word count");
        job.setJarByClass(WordCount.class);
        job.setMapperClass(TokenizerMapper.class);
        job.setReducerClass(IntSumReducer.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);
        for (int i = 0; i < otherArgs.length - 1; ++i) {
            FileInputFormat.addInputPath(job, new Path(otherArgs[i]));
        }
        FileOutputFormat.setOutputPath(job,
                new Path(otherArgs[otherArgs.length - 1]));
        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
```

ANNEXE : 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 http://maven.apache.org/xsd
    <modelVersion>4.0.0</modelVersion>

    <groupId>dle.wc</groupId>
    <artifactId>HadoopWordCount</artifactId>
    <version>1.0-SNAPSHOT</version>

    <properties>
        <maven.compiler.source>8</maven.compiler.source>
        <maven.compiler.target>8</maven.compiler.target>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.apache.hadoop</groupId>
            <artifactId>hadoop-core</artifactId>
            <version>1.2.1</version>
        </dependency>
        <dependency>
            <groupId>org.apache.hadoop</groupId>
            <artifactId>hadoop-common</artifactId>
            <version>3.4.2</version>
        </dependency>
    </dependencies>
</project>
```