

Nama : Annisya Ekapratiwi Aprilia
NIM : 221011036
Kelas : IK22-A

Laporan Praktikum 1

Tutorial Instalasi Hadoop

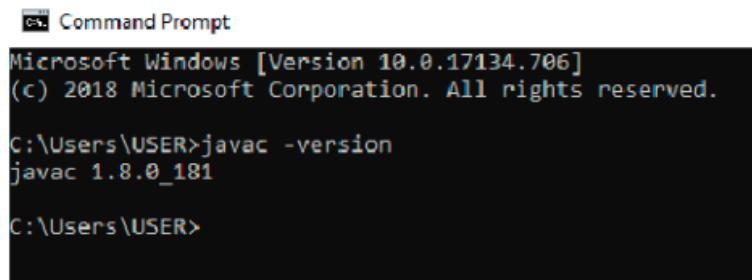
1. Langkah pertama Download Hadoop di Google

<https://hadoop.apache.org/releases.html>

2. Kemudian Download JDK yang kompatibel dengan hadoop

<https://www.oracle.com/id/java/technologies/downloads/>

3. Periksa apakah java sudah diinstal pada sistem atau tidak, gunakan “Java –version” untuk memeriksa. Jika java belum diinstal pada sistem, maka anda harus menginstall java terlebih dahulu

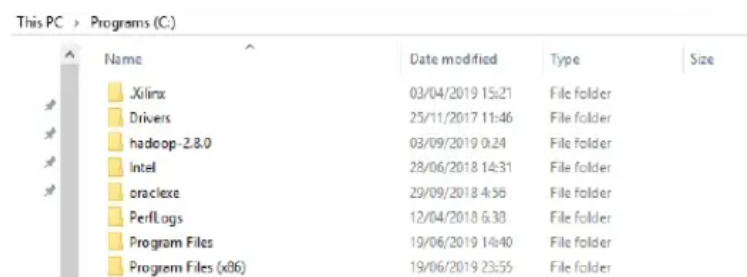


```
Microsoft Windows [Version 10.0.17134.706]
(c) 2018 Microsoft Corporation. All rights reserved.

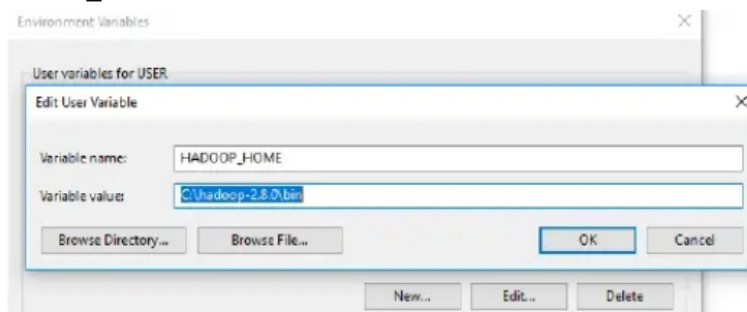
C:\Users\USER>javac -version
javac 1.8.0_181

C:\Users\USER>
```

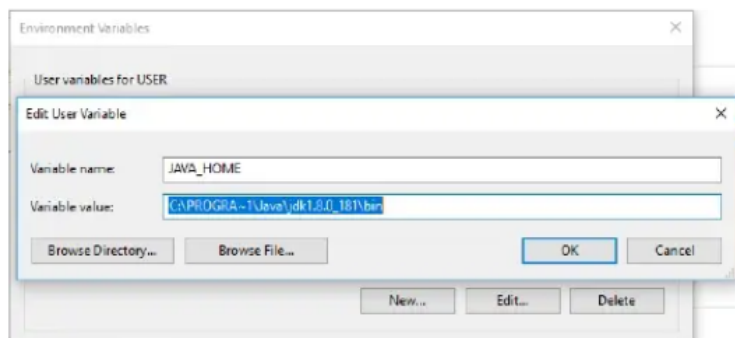
4. Ekstrak file Hadoop.zip dan letakkan di C:\Hadoop”



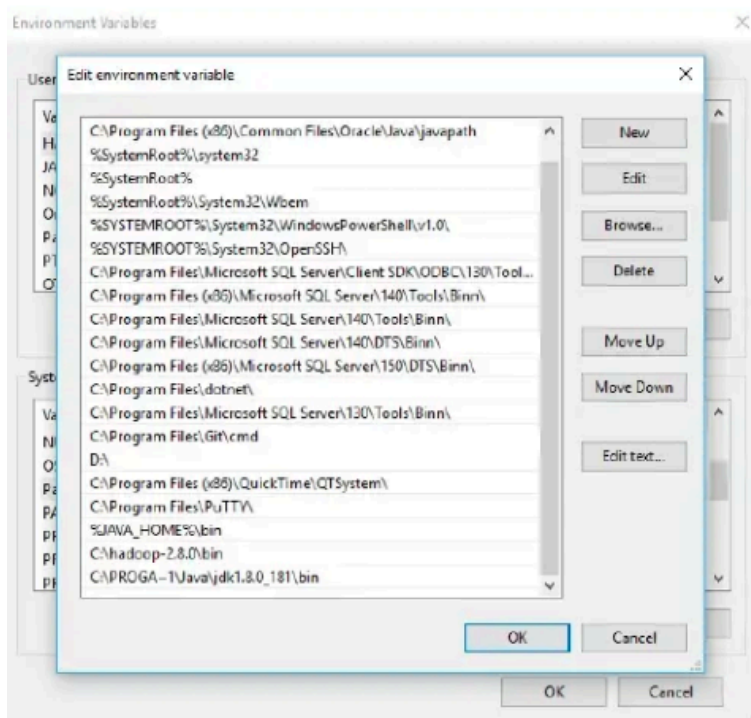
5. Settings path HADOOP_HOME environment variable.



6. Setting path JAVA_HOME environment variable



7. Setting Hadoop bin directory path dan JAVA bin directory path.



Konfigurasi

1. Edit file C:/Hadoop-2.8.0/etc/hadoop/core-site.xml, salin di bawah xml paragraf kemudian simpan file.

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

2. Ubah nama mapred-site.xml.template menjadi mapred-site.xml dan edit file C:/Hadoop-2.8.0/etc/hadoop/mapred-site.xml, salin di bawah xml paragraf kemudian simpan file.

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

3. Buat folder data di C:\Hadoop-2.8.0
 - a. Buat folder “datanode” di C:\Hadoop-2.8.0\data
 - b. Buat folder “namenode” di C:\Hadoop-2.8.0\data

Programs (C:) > hadoop-2.8.0

Name	Date modified	Type	Size
bin	03/09/2019 0:24	File folder	
data	03/09/2019 0:53	File folder	
etc	03/09/2019 0:24	File folder	
include	03/09/2019 0:24	File folder	
lib	03/09/2019 0:24	File folder	
libexec	03/09/2019 0:24	File folder	
sbin	03/09/2019 0:24	File folder	
share	03/09/2019 0:39	File folder	
LICENSE	17/03/2017 12:31	Text Document	97 KB
NOTICE	17/03/2017 12:31	Text Document	16 KB
README	17/03/2017 12:31	Text Document	2 KB

4. Edit file C:\Hadoop-2.8.0/etc/hadoop/hdfs-site.xml, salin di bawah xml paragraf kemudian simpan file.

```
<configuration>
  <property>
    <name>dfs.replication</name>
    <value>1</value>
  </property>
  <property>
    <name>dfs.namenode.name.dir</name>
    <value>C:\hadoop-2.8.0\data\namenode</value>
  </property>
  <property>
    <name>dfs.datanode.data.dir</name>
    <value>C:\hadoop-2.8.0\data\datanode</value>
  </property>
</configuration>
```

5. Edit file C:\Hadoop-2.8.0/etc/hadoop/yarn-site, salin di bawah xml paragraf kemudian simpan file

```
<configuration>
  <property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
  </property>
  <property>
    <name>yarn.nodemanager.auxservices.mapreduce.shuffle.class</name>
    <value>org.apache.hadoop.mapred.ShuffleHandler</value>
  </property>
</configuration>
```

6. Edit file C:\Hadoop-2.8.0/etc/hadoop/hadoop-env.cmd dengan menutup command line “JAVA_HOME=%JAVA_HOME%” kemudian ganti dengan set JAVA_HOME=C:\PROGRA~1\Java\jdk1.8.0_181 ini adalah path untuk menuju file jdk.

```
rem The Java implementation to use. Required.
rem set JAVA_HOME=%JAVA_HOME%
set JAVA_HOME=C:\PROGRA~1\Java\jdk1.8.0_181
```

Konfigurasi Hadoop

1. Download file hadoop Konfigurasi.zip

- [illegible]

1. Open cmd dan ganti direktori ke “C:\Hadoop-2.8.0\sbin” kemudian ketik”start-all.cmd” untuk memulai apache.

```
C:\ Command Prompt
Microsoft Windows [Version 10.0.17134.706]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\USER>cd..

C:\Users>cd..

C:\>cd Hadoop-2.8.0\sbin

C:\hadoop-2.8.0\sbin>start-all.cmd
This script is Deprecated. Instead use start-dfs.cmd and start-yarn.cmd
starting yarn daemons

C:\hadoop-2.8.0\sbin>
```

- Hadoop Namenode
- Hadoop datanode
- YARN Resourc manager
- Yarn Node Manager

[illegible]

3. Buka localhost:8088 untuk membuka Resource Manager

Cluster

About Nodes Node Labels Assemblies NEW NEW SAVING SUBMITTED ACCEPTED RUNNING FINISHED FAILED SUCCEEDED Scheduler Tools

Cluster Metrics

Apps Submitted	0	Apps Pending	0	Apps Running	0	Apps Completed	0	Containers Running	0	Used Resources	memory 0 B, vCores 0	Total RB	
Cluster Nodes Metrics													
Active Nodes				Decommissioning Nodes				Decommissioned Nodes				Lost Nodes	
0				0				0				0	
Scheduler Metrics													
Capacity Scheduler													
Scheduler Type													
Scheduling Resource Type													
Minimum Allocation													
Maximum Allocation													
Show 20 settings													
ID	User	Name	Application Type	Application Tags	Queue	Application Priority	StartTime	LaunchTime	FinishTime	State	FinalStatus	Running Containers	Allocated CPU V-Cores
No data available in table													

Thursday, January 2, 2025

4. Buka localhost:9870 untuk checkout the health of Name Node

Hadoop Overview Datanodes Datanode Volumes Failures Snapshot Startup Progress Utilities

Overview 'localhost:9000' (v-active)

Started:

Thu Jan 02 21:09:29 +0800 2025

Version:

3.3.6 (r1b70238729a6006a4f8b19c556808d0124dc)

Compiled:

Sun Jan 18 16:22:00 +0800 2023 by ubuntu from (HEAD detached at release-3.3.6-RC1)

Cluster ID:

CD-258d9fa-69a7-4aaa-b015-6b2b5a08c44

Block Pool ID:

BP-1493234063-192-188-1735622898054

Summary

Security is off.
SafeMode is off.
1 files and directories, 0 blocks (0 replicated blocks, 0 ensure coded block groups) = 1 total Hadoop object(s).
Heap Memory used 52.37 MB of 96 MB Heap Memory. Max Heap Memory is 1000 MB.
Non Heap Memory used 60.28 MB of 63.27 MB Committed Non Heap Memory. Max Non Heap Memory is 'unbounded'.
Configured Capacity: 0 B
Configured Remote Capacity: 0 B

Laporan Praktikum 2

1. Demo HDFS

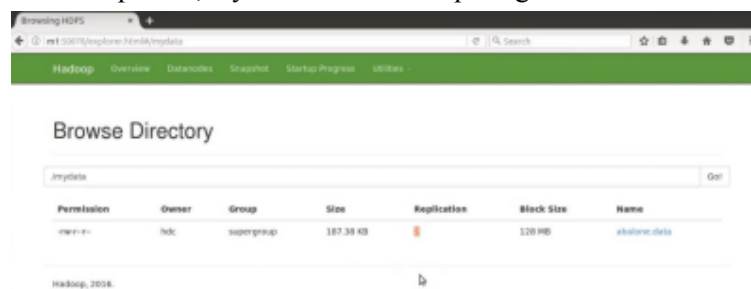
Dalam demo ini, kita akan melihat perintah-perintah yang akan membantu kita menuliskan data ke suatu cluster dua-node yang memiliki dua DataNodes, dua NodeManager, dan satu mesin Master. Ada tiga cara untuk menulis data:

- Melalui perintah langsung (command line).
- Dengan menuliskan kode program.
- Menggunakan suatu Graphical User interface (GUI).

Sebelum dimulai, kita harus mengunduh beberapa dataset sampel yang akan digunakan untuk menulis data ke dalam HDFS. Sekarang, kita akan melihat cara menjalankan beberapa perintah pada HDFS. Ini adalah perintah yang dapat dimulai:

- `hdfs dfs -mkdir /mydata` // To create a directory on HDFS
- `ls` // This lists down the files
- `hdfs dfs -copyFromLocal aba* /mydata` // Copies file from local file system to HDFS
- `hdfs dfs -ls /mydata` // Lists the directory

Setelah perintah ini, menggunakan antarmuka web, kita dapat memeriksa apakah file telah direplikasi. Jika direplikasi, layar akan terlihat seperti gambar berikut:



Sekarang, mari kita berikan perintah tambahan:

```
cp hadoop-hdc-datanode-m1.log cp hadoop-hdc-datanode-m2.log  
cp hadoop-hdc-datanode-m1.log cp hadoop-hdc-datanode-m3.log  
cp hadoop-hdc-datanode-m1.log cp hadoop-hdc-datanode-m3.log
```

// Above commands creates multiple files

`hdfs dfs -mkdir /mydata2` // Creates a new directory on HDFS

`hdfs dfs -put hadoop-hdc-datanode-m* /mydata2` // Copies multiple files

`hdfs dfs -setrep -R -w 2 /mydata2` // Sets replication factor to 2

`hdfs dfs -rm -R /mydata2` // Removes data from HDFS

2. Demo MapReduce

Dalam demo MapReduce ini, kita akan melihat cara mendapatkan jumlah total URL yang paling sering dikunjungi. Pertama, kita harus menggunakan file sampel yang memiliki daftar URL yang kemudian kita hitung kemunculan totalnya. Sampel dataset dapat berasal dari log proxy server atau web server. Untuk mengimplementasikan program ini menggunakan pendekatan MapReduce, silakan ikuti langkah-langkah ini:

- a. Gunakan program Mapper di bawah ini yang ditulis untuk melakukan tugas map:

```
1 package org.example.HCodes;
2
3 import java.io.IOException;
4 import java.util.logging.Level;
5 import java.util.logging.Logger;
6 import org.apache.hadoop.mapreduce.Mapper;
7 import org.apache.hadoop.io.IntWritable;
8 import org.apache.hadoop.io.Text;
9
10 public class URLCountM extends Mapper<Text, Text, Text, IntWritable> {
11     private static final Logger LOG = Logger.getLogger(URLCountM.class.getName());
12
13     public final IntWritable iw = new IntWritable();
14
15     @Override
16     public void map(Text key, Text value, Context context) {
17         try {
18             LOG.log(Level.INFO, "MAPPER_KEY: " + key.toString().concat(" MAPPER_VALUE: " + value.toString());
19             context.write(key, new IntWritable(Integer.valueOf(value.toString())));
20         } catch (NumberFormatException | IOException | InterruptedException e) {
21             LOG.log(Level.SEVERE, "ERROR: " + e.toString());
22         }
23     }
24 }
25
26 }
```

- b. Gunakan program Reducer di bawah ini untuk melakukan agregasi.

```
1 package org.example.HCodes;
2
3 import java.io.IOException;
4
5 public class URLCountR extends Reducer<Text, IntWritable, Text, IntWritable> {
6     private static final Logger LOG = Logger.getLogger(URLCountR.class.getName());
7
8     private IntWritable result = new IntWritable();
9
10    @Override
11    public void reduce(Text key, Iterable<IntWritable> values, Context context)
12        throws IOException, InterruptedException {
13        int sum = 0;
14        for (IntWritable val : values) {
15            sum += val.get();
16        }
17        result.set(sum);
18        LOG.log(Level.INFO, "REDUCER_VALUE: " + result.toString());
19        context.write(key, result);
20    }
21 }
22
23 }
```

- c. Gunakan program driver di bawah ini untuk memahami klas mapper, kelas reducer, format kunci output dan format nilai

```
1 package org.example.HCodes;
2
3 import org.apache.hadoop.conf.Configuration;
4
5 public class URLCount extends Configured implements Tool {
6
7     public static void main(String[] args) throws Exception {
8         int res = ToolRunner.run(new Configuration(), new URLCount(), args);
9         System.exit(res);
10    }
11
12    @Override
13    public int run(String[] args) throws Exception {
14        Configuration conf = this.getConf();
15        conf.set("mapreduce.input.keyvaluelinerecordreader.key.value.separator", " ");
16        Job job = Job.getInstance(conf, "URLCount");
17        job.setJarByClass(getClass());
18        job.setInputFormatClass(KeyValueTextInputFormat.class);
19        job.setOutputFormatClass(TextOutputFormat.class);
20        job.setMapperClass(URLCountM.class);
21        job.setReducerClass(URLCountR.class);
22        job.setMapOutputKeyClass(Text.class);
23        job.setMapOutputValueClass(IntWritable.class);
24        job.setOutputKeyClass(IntWritable.class);
25        job.setOutputValueClass(Text.class);
26        FileInputFormat.addInputPath(job, new Path(args[0]));
27    }
28 }
```



```

URLCount.java
17
18 public static void main(String[] args) throws Exception {
19
20     int res = ToolRunner.run(new Configuration(), new URLCount(), args);
21     System.exit(res);
22
23 }
24
25 @Override
26 public int run(String[] args) throws Exception {
27     Configuration conf = this.getConf();
28     conf.set("mapreduce.input.keyvaluelinerecordreader.key.value.separator", " ");
29     Job job = Job.getInstance(conf, "URLCount");
30     job.setJarByClass(getClass());
31     job.setInputFormatClass(KeyValueTextInputFormat.class);
32     job.setOutputFormatClass(TextOutputFormat.class);
33     job.setMapperClass(URLCountM.class);
34     job.setReducerClass(URLCountR.class);
35     job.setMapOutputKeyClass(Text.class);
36     job.setMapOutputValueClass(IntWritable.class);
37     job.setOutputKeyClass(IntWritable.class);
38     job.setOutputValueClass(Text.class);
39     FileInputFormat.addInputPath(job, new Path(args[0]));
40     FileOutputFormat.setOutputPath(job, new Path(args[1]));
41     return (job.waitForCompletion(true) == true ? 0 : -1);
42 }
43
44 }

```

- d. Setelah menulis kode, kita dapat mengekspor ini ke bwnuk file jar. Perlu ada file dalam HDFS untuk melaksanakan MapReduce. untk itu,kita harus masuk(log in) ke cluster tersebut terlebih dahulu.
- e. Masukkan file sampel (file yang memiliki URL) ke HDFS menggunakan perintah -put.
- f. Menggunakan perintah di bawah ini, kita akan menjalankan program MapReduce dan mendapatkan jumlah kumulatif berapa kali suatu URL dikunjungi.
`hadoop jar URLCount.jar org.example.HCodes.URLCount /user/(mention directory where the input is present) /user/(mention directory where the output should be seen - destination path)`
- g. Setelah menjalankan kode di atas, kita akan melihat bahwa pekerjaan MapReduce dikirimkan ke cluster YARN. Setiap kali program MapReduce berjalan, kita akan memiliki satu atau lebih file bagian yang dibuat sebagai output. Dalam hal ini, Kita memiliki satu tugas map dan satu tugas reduce, dan karenanya jumlah file bagian juga akan menjadi satu. Kode berikut digunakan untuk menampilkan jumlah file bagian dan hasil akhir:
 - `hdfs dfs -ls /user(give your destination path) //Displays the number of part files`
 - `hdfs dfs -cat /user(give your destination path)/part-r-00000(mention the part details) //Displays the final output`

3. Demo YARN

Di bawah ini adalah beberapa perintah YARN yang paling sering digunakan:

- `yarn version //Displays the Hadoop and vendor-specific distribution version`
- `yarn application -list //Lists all the applications running`
- `yarn application -list -appSTATES -FINISHED //Lists the services that are finished running`
- `yarn application -status give application ID //Prints the status of the applications`
- `yarn application -kill give application ID //Kills a running application`
- `yarn node -list //Gives the list of node managers`
- `yarn rmadmin -getGroups hdfs //Gives the group HDFS belongs to`