Principles of HDFS & MapReduce2:

Name: อวยชัย   ภิรมย์รื่น

Tel.   : 086-813-5354

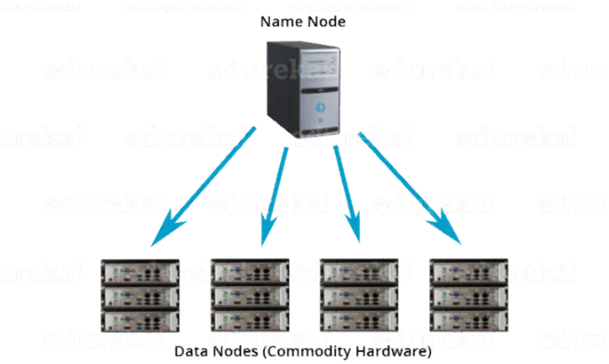
e-mail : p.Auoychai@gmail.com

Wrap up

Principles of HDFS :

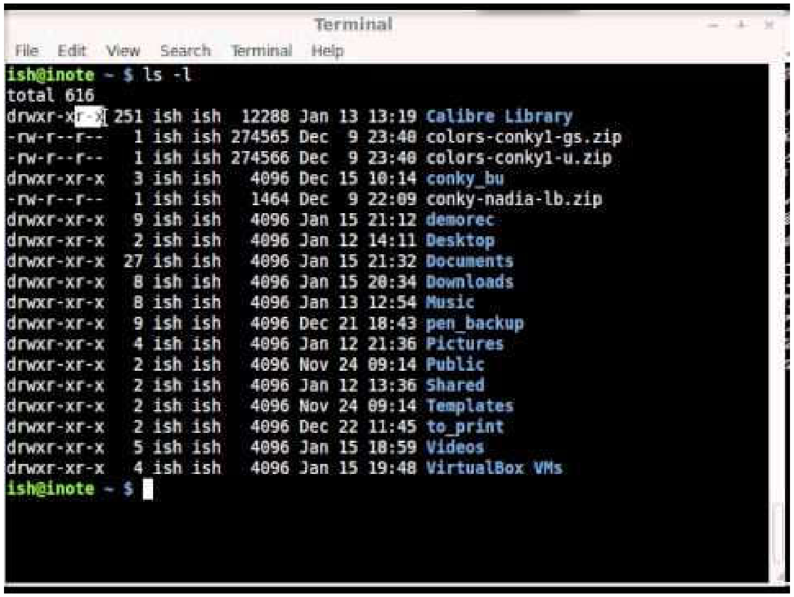
# HDFS Architecture :

- **What is HDFS** , Java based distributed file system and allow to store large data across multiple nodes in a Hadoop Cluster.



Distributed Storage:

Can't use Linux command to see it



```
ish@inote ~ $ ls -l
total 616
drwxr-xr-x 251 ish ish  12288 Jan 13 13:19 Calibre Library
-rw-r--r--   1 ish ish 274565 Dec  9 23:40 colors-conky1-gs.zip
-rw-r--r--   1 ish ish 274566 Dec  9 23:40 colors-conky1-u.zip
drwxr-xr-x   3 ish ish   4096 Dec 15 10:14 conky_bu
-rw-r--r--   1 ish ish   1464 Dec  9 22:09 conky-nadia-lb.zip
drwxr-xr-x   9 ish ish   4096 Jan 15 21:12 demorec
drwxr-xr-x   2 ish ish   4096 Jan 12 14:11 Desktop
drwxr-xr-x  27 ish ish   4096 Jan 15 21:32 Documents
drwxr-xr-x   8 ish ish   4096 Jan 15 20:34 Downloads
drwxr-xr-x   8 ish ish   4096 Jan 13 12:54 Music
drwxr-xr-x   9 ish ish   4096 Dec 21 18:43 pen_backup
drwxr-xr-x   4 ish ish   4096 Jan 12 21:36 Pictures
drwxr-xr-x   2 ish ish   4096 Nov 24 09:14 Public
drwxr-xr-x   2 ish ish   4096 Jan 12 13:36 Shared
drwxr-xr-x   2 ish ish   4096 Nov 24 09:14 Templates
drwxr-xr-x   2 ish ish   4096 Dec 22 11:45 to_print
drwxr-xr-x   5 ish ish   4096 Jan 15 18:59 Videos
drwxr-xr-x   4 ish ish   4096 Jan 15 19:48 VirtualBox VMs
ish@inote ~ $
```

MasterNode:

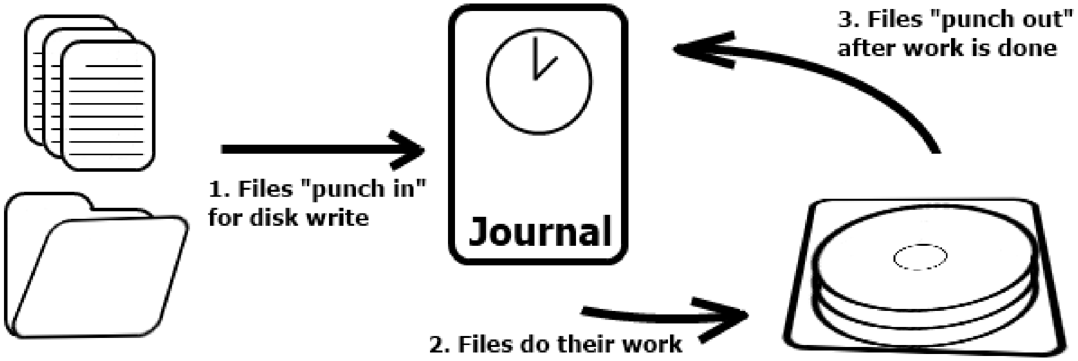
NameNode , ResourceManager

SlaveNode:

DataNode , NodeManager

# HDFS Write-Read Process :

What the HDFS difference than traditional File System



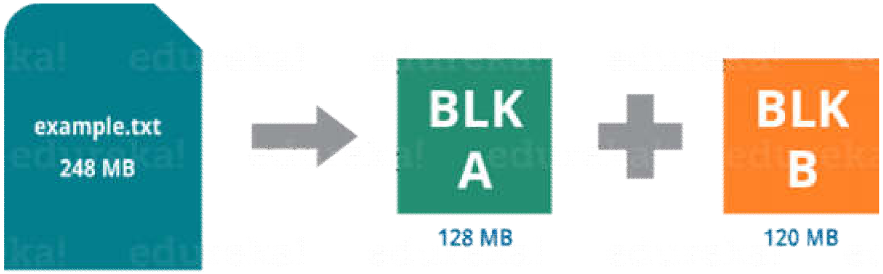
https://www.howtogeek.com/wp-content/uploads/2010/12/640x250xJournal.png.pagespeed.gp+jp+jw+pj+js+rj+rp+rw+ri+cp+md.ic.4h1Q9x_447.png

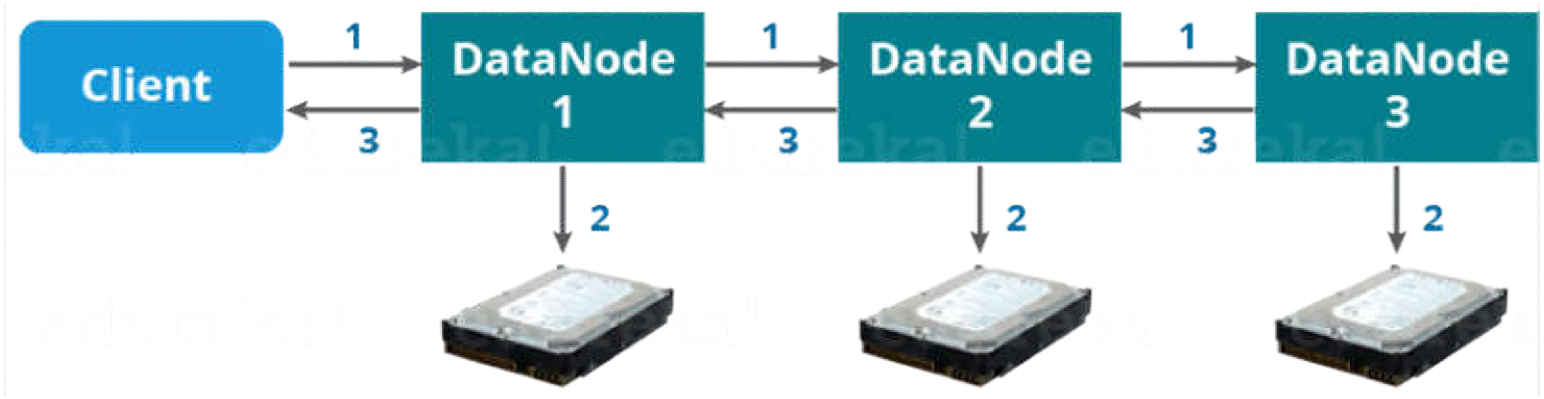
# **HDFS Write Process :**

1.Set up of Pipeline

2.Data streaming and replication

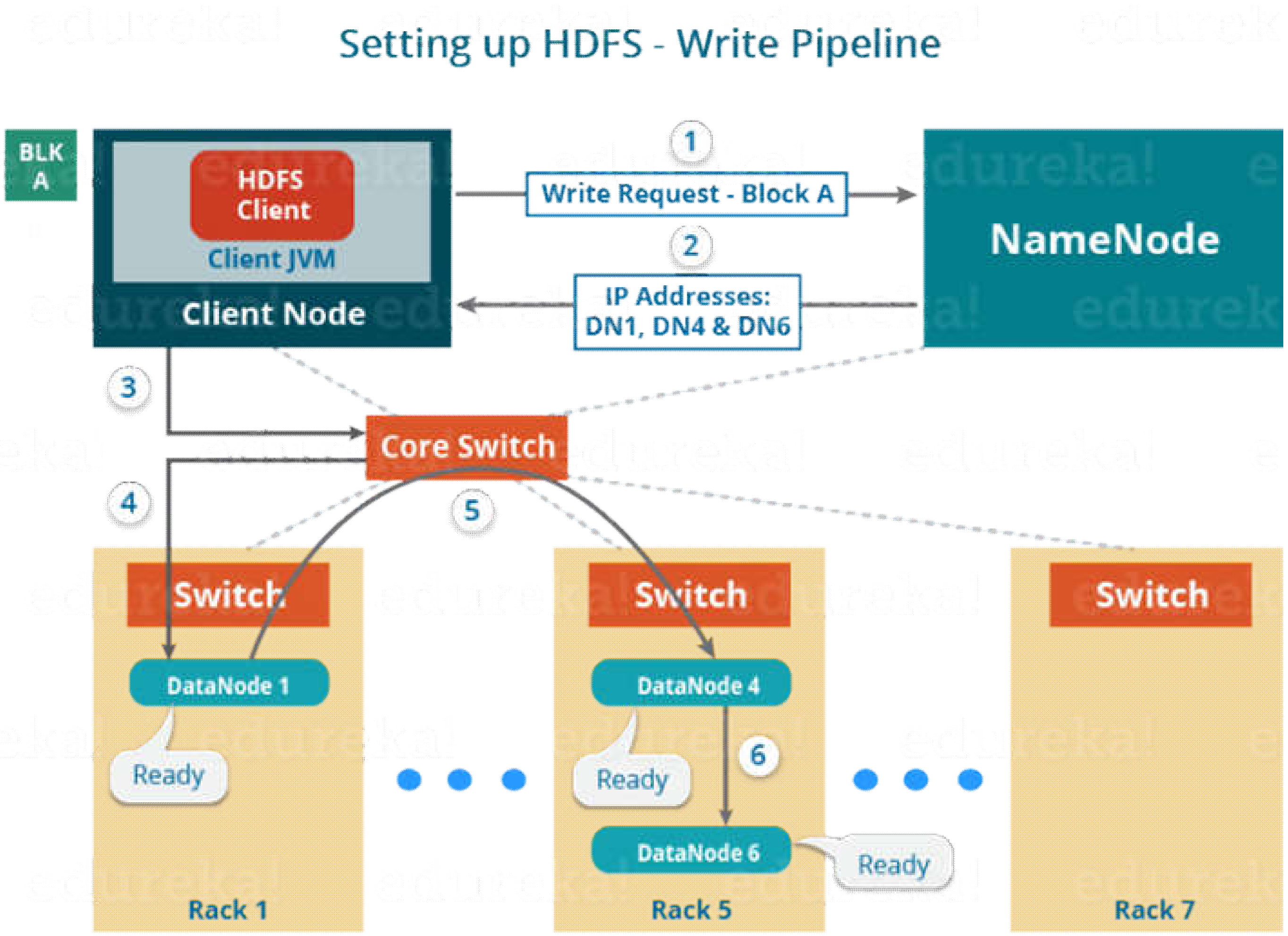
3.Shutdown of Pipeline (Acknowledgement stage)





# HDFS Write Process :

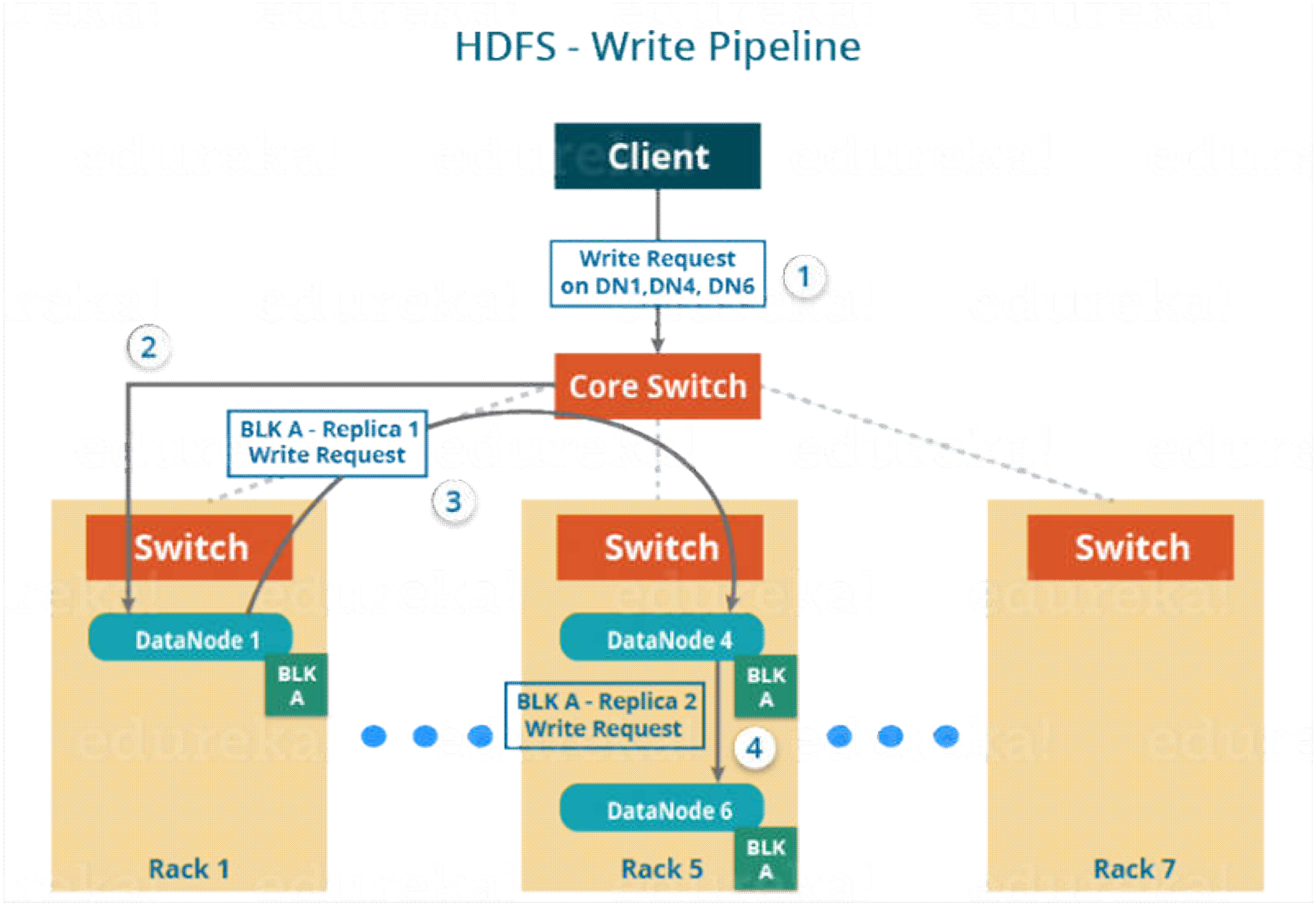
1). Set up of Pipeline



https://www.edureka.co

# HDFS Write Process :

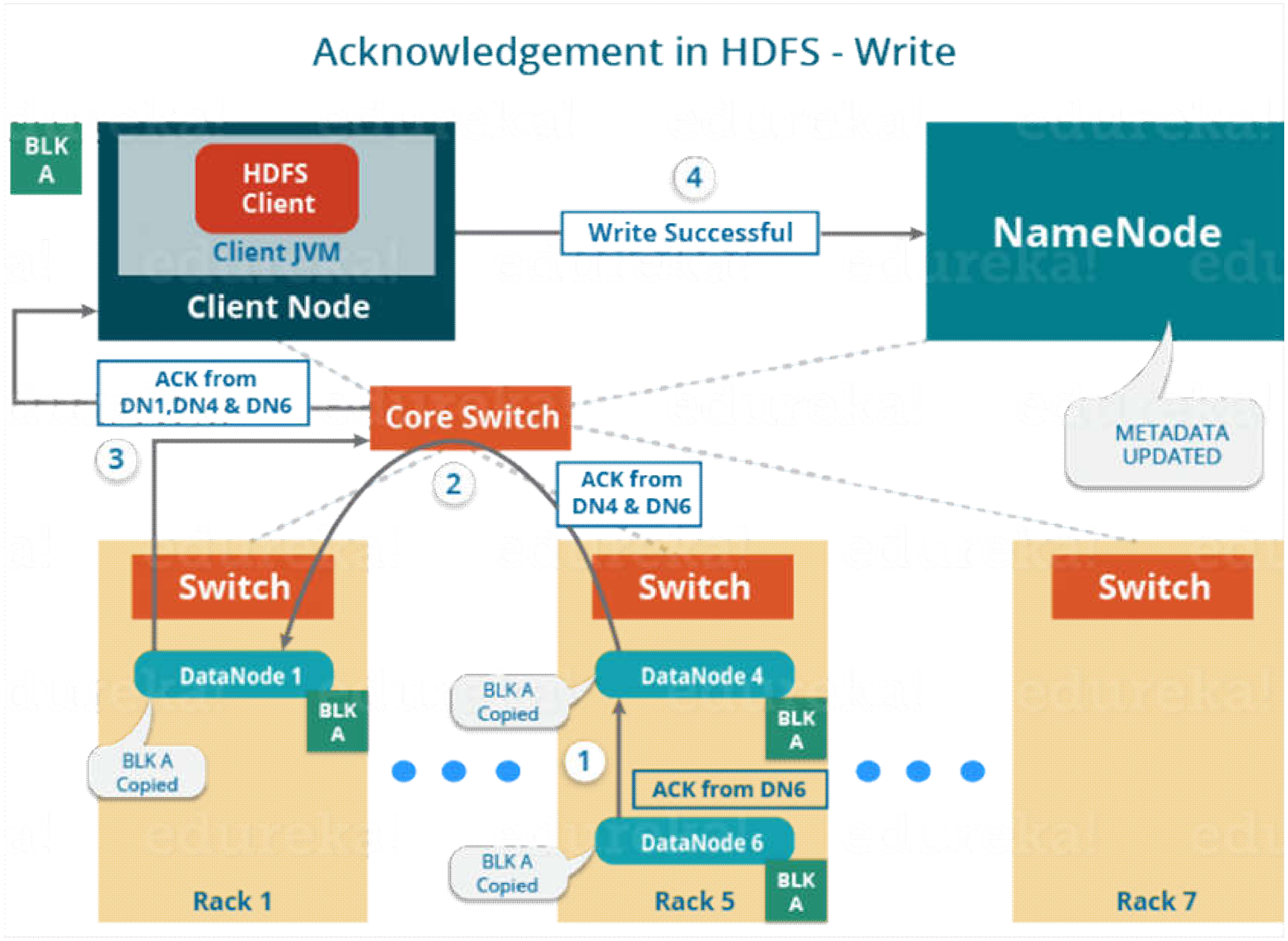
2). Data streaming and replication



https://www.edureka.co

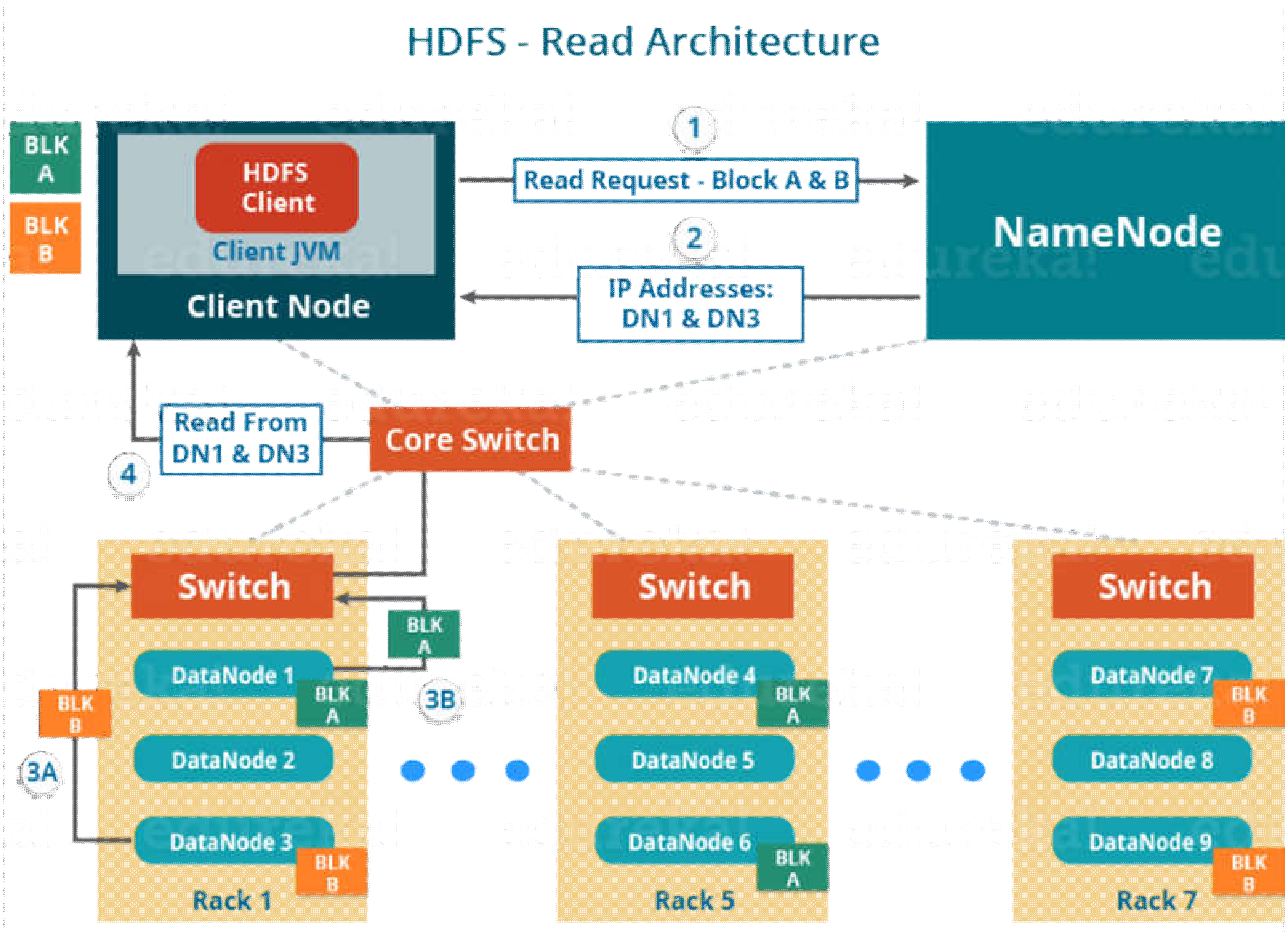
HDFS Write Process :

3). Shutdown of Pipeline



https://www.edureka.co

# HDFS Read Process :

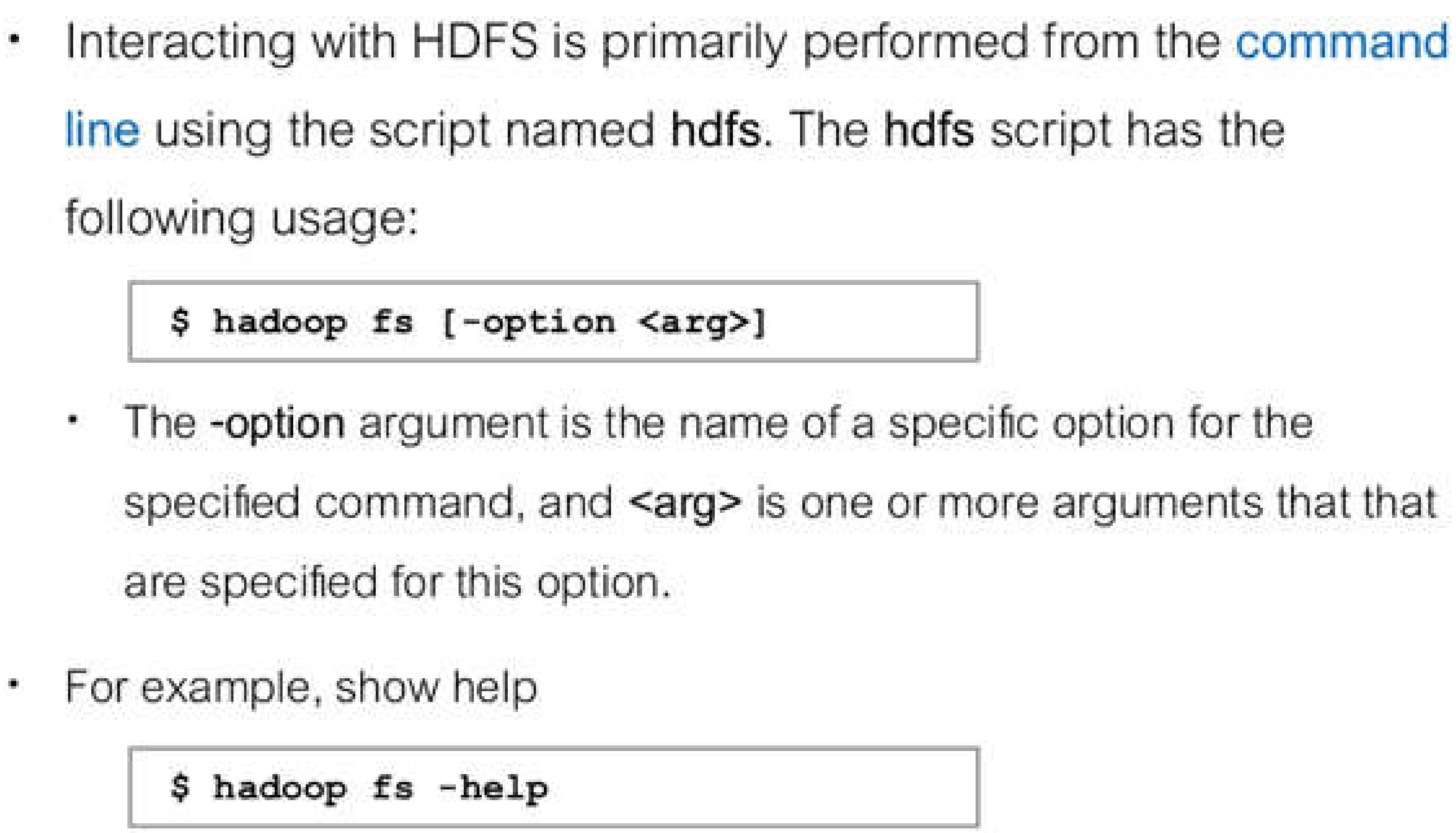


https://www.edureka.co

# **Hadoop Command :**

- File System

# Interact to HDFS :



- Interacting with HDFS is primarily performed from the command line using the script named **hdfs**. The **hdfs** script has the following usage:

```
$ hadoop fs [-option <arg>]
```

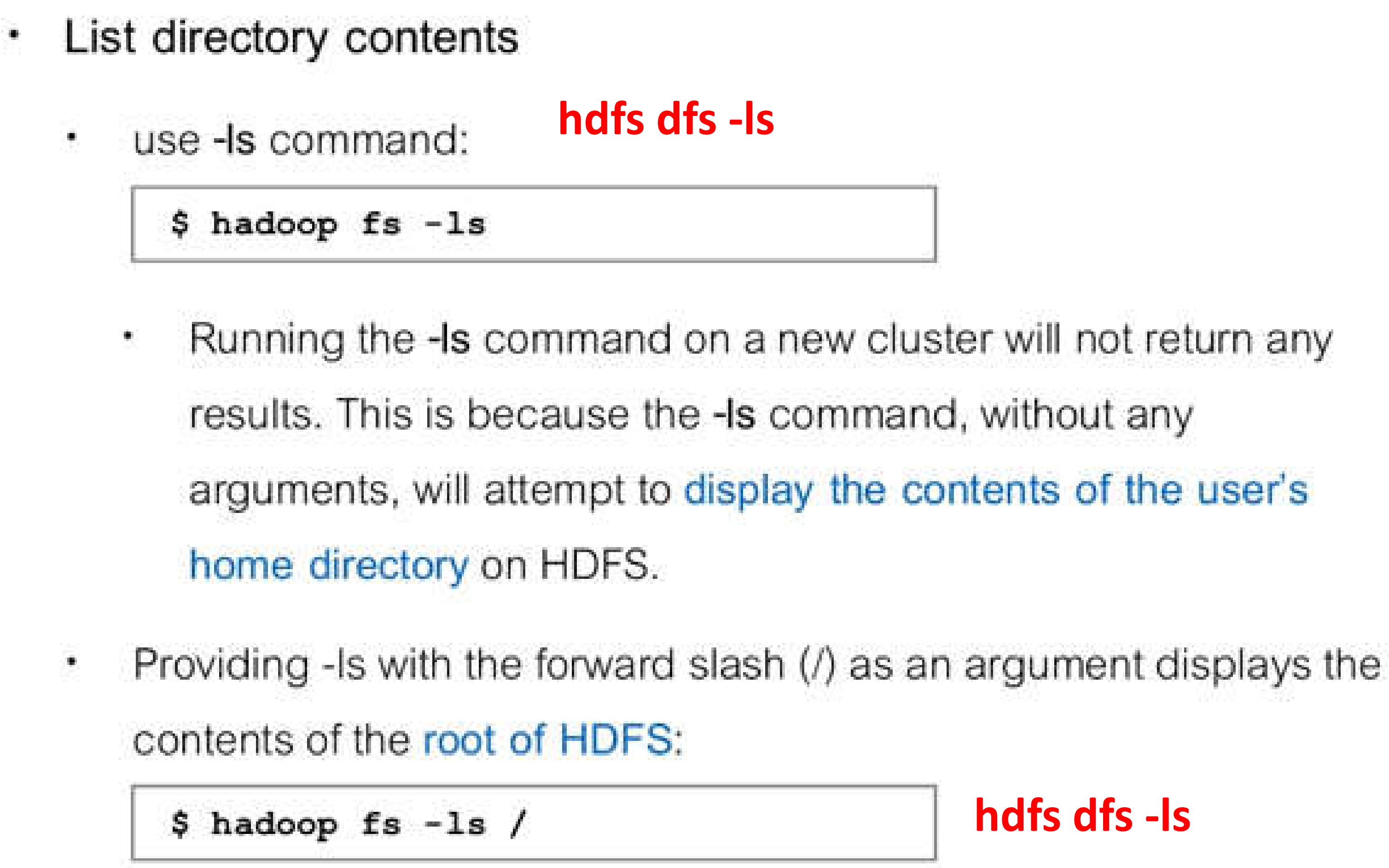
  - The **-option** argument is the name of a specific option for the specified command, and **<arg>** is one or more arguments that that are specified for this option.

- For example, show help

```
$ hadoop fs -help
```

https://www.slideshare.net/eakasit_dpu/introduction-to-hadoop-and-mapreduce-75323926?qid=6dc387d1-f614-4646-a2cb-b4805805235a&v=&b=&from_search=3
Dr. Eakasit Pacharawongsakda @dpu.ac.th

Interact to HDFS :



- List directory contents
  - use -ls command:

hdfs dfs -ls

```
$ hadoop fs -ls
```

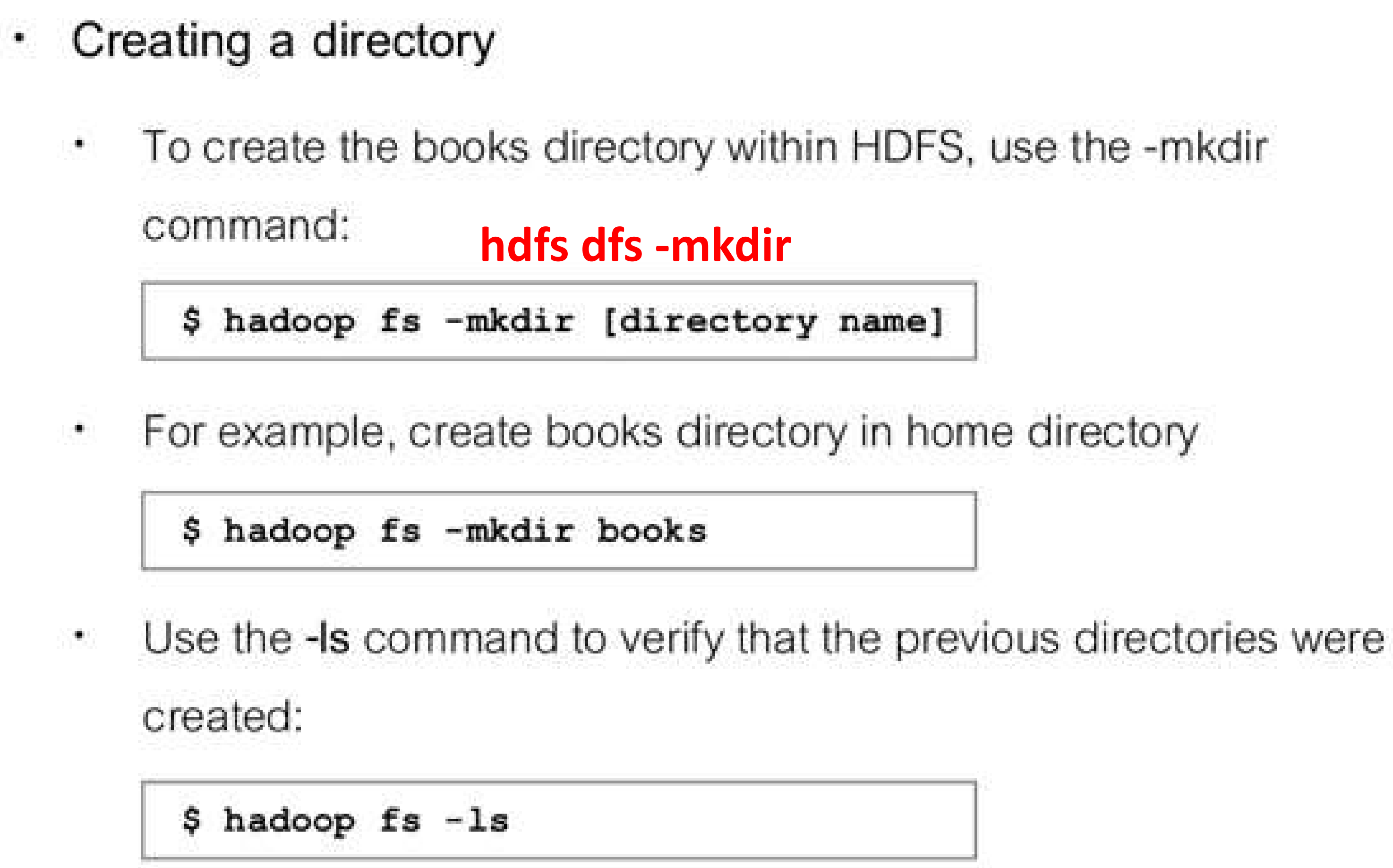
    - Running the -ls command on a new cluster will not return any results. This is because the -ls command, without any arguments, will attempt to display the contents of the user's home directory on HDFS.
  - Providing -ls with the forward slash (/) as an argument displays the contents of the root of HDFS:

```
$ hadoop fs -ls /
```

hdfs dfs -ls

https://www.slideshare.net/eakasit_dpu/introduction-to-hadoop-and-mapreduce-75323926?qid=6dc387d1-f614-4646-a2cb-b4805805235a&v=&b=&from_search=3

# Interact to HDFS :



- Creating a directory
  - To create the books directory within HDFS, use the -mkdir command:

hdfs dfs -mkdir

```
$ hadoop fs -mkdir [directory name]
```

  - For example, create books directory in home directory

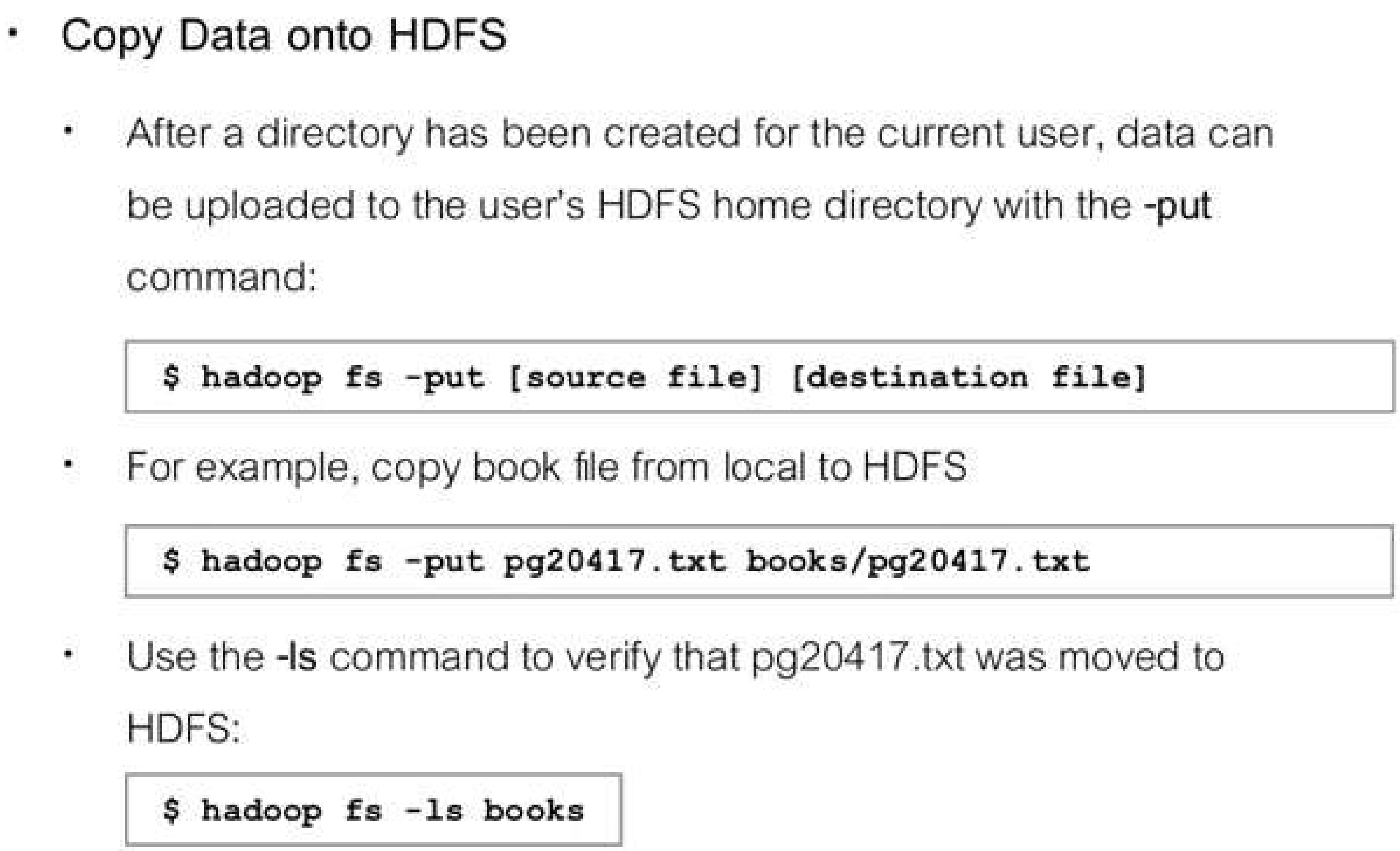
```
$ hadoop fs -mkdir books
```

  - Use the -ls command to verify that the previous directories were created:

```
$ hadoop fs -ls
```

https://www.slideshare.net/eakasit_dpu/introduction-to-hadoop-and-mapreduce-75323926?qid=6dc387d1-f614-4646-a2cb-b4805805235a&v=&b=&from_search=3

# **Interact to HDFS** :



- Copy Data onto HDFS
  - After a directory has been created for the current user, data can be uploaded to the user's HDFS home directory with the **-put** command:

```
$ hadoop fs -put [source file] [destination file]
```

  - For example, copy book file from local to HDFS

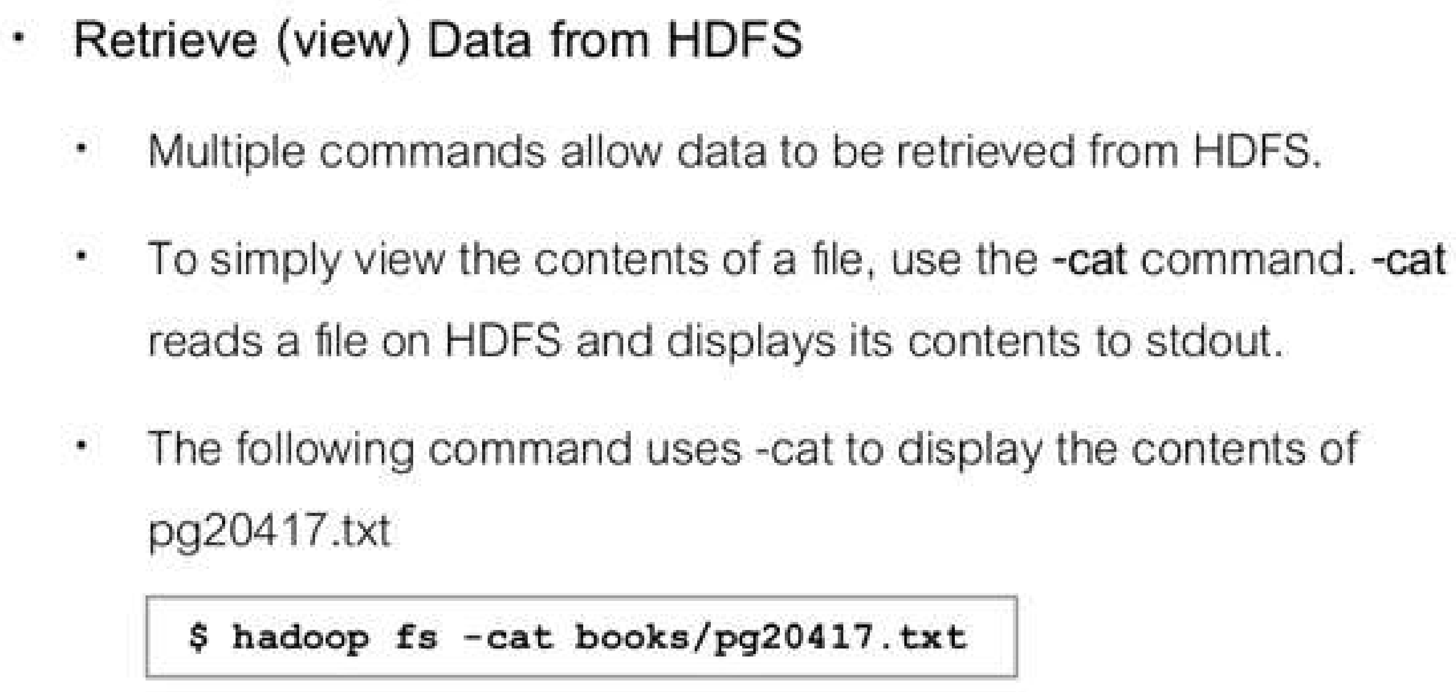
```
$ hadoop fs -put pg20417.txt books/pg20417.txt
```

  - Use the **-ls** command to verify that pg20417.txt was moved to HDFS:

```
$ hadoop fs -ls books
```

https://www.slideshare.net/eakasit_dpu/introduction-to-hadoop-and-mapreduce-75323926?qid=6dc387d1-f614-4646-a2cb-b4805805235a&v=&b=&from_search=3

# Interact to HDFS :

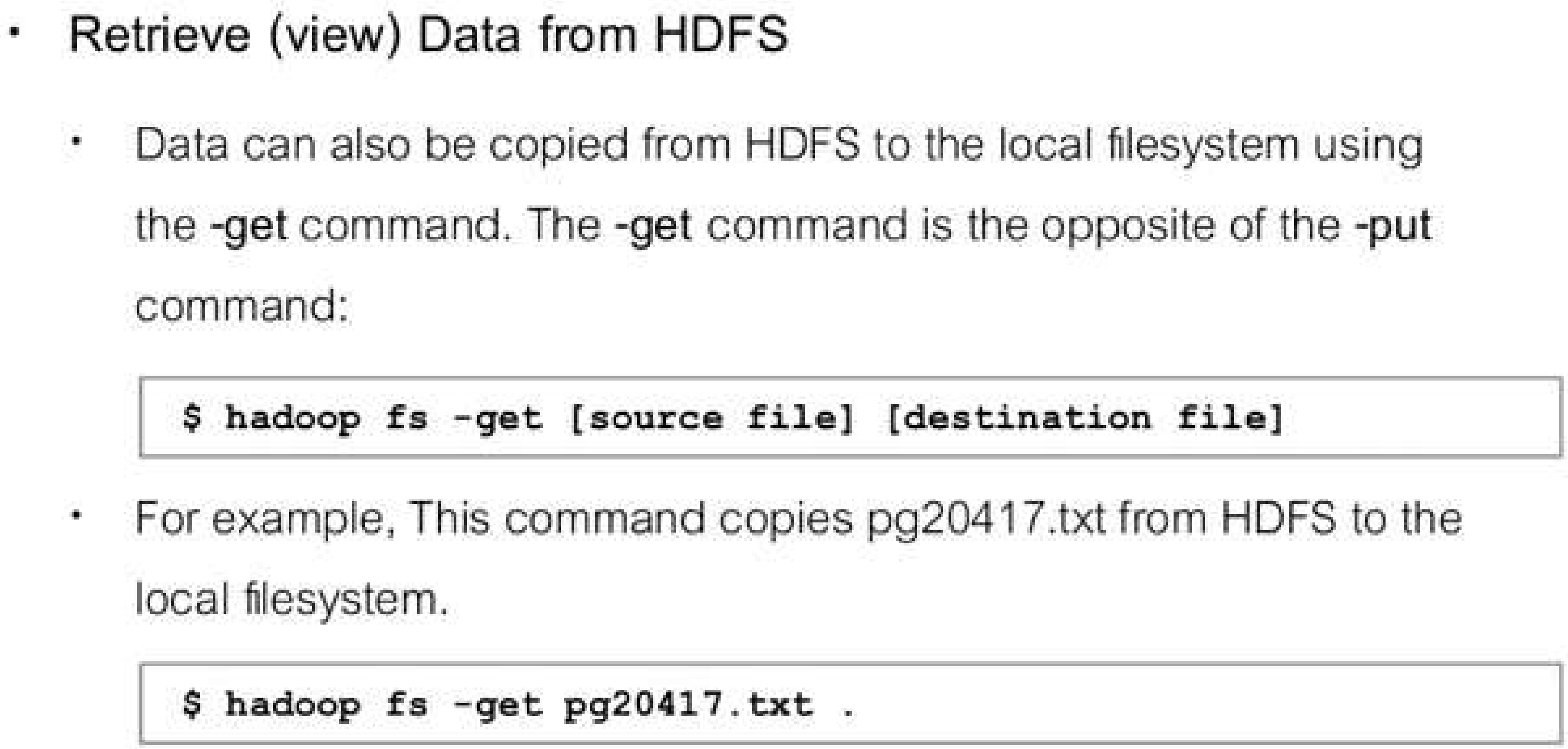


- Retrieve (view) Data from HDFS
  - Multiple commands allow data to be retrieved from HDFS.
  - To simply view the contents of a file, use the **-cat** command. **-cat** reads a file on HDFS and displays its contents to stdout.
  - The following command uses -cat to display the contents of pg20417.txt

```
$ hadoop fs -cat books/pg20417.txt
```

https://www.slideshare.net/eakasit_dpu/introduction-to-hadoop-and-mapreduce-75323926?qid=6dc387d1-f614-4646-a2cb-b4805805235a&v=&b=&from_search=3

# Interact to HDFS :



- Retrieve (view) Data from HDFS
  - Data can also be copied from HDFS to the local filesystem using the **-get** command. The **-get** command is the opposite of the **-put** command:

```
$ hadoop fs -get [source file] [destination file]
```

  - For example, This command copies pg20417.txt from HDFS to the local filesystem.

```
$ hadoop fs -get pg20417.txt .
```

https://www.slideshare.net/eakasit_dpu/introduction-to-hadoop-and-mapreduce-75323926?qid=6dc387d1-f614-4646-a2cb-b4805805235a&v=&b=&from_search=3

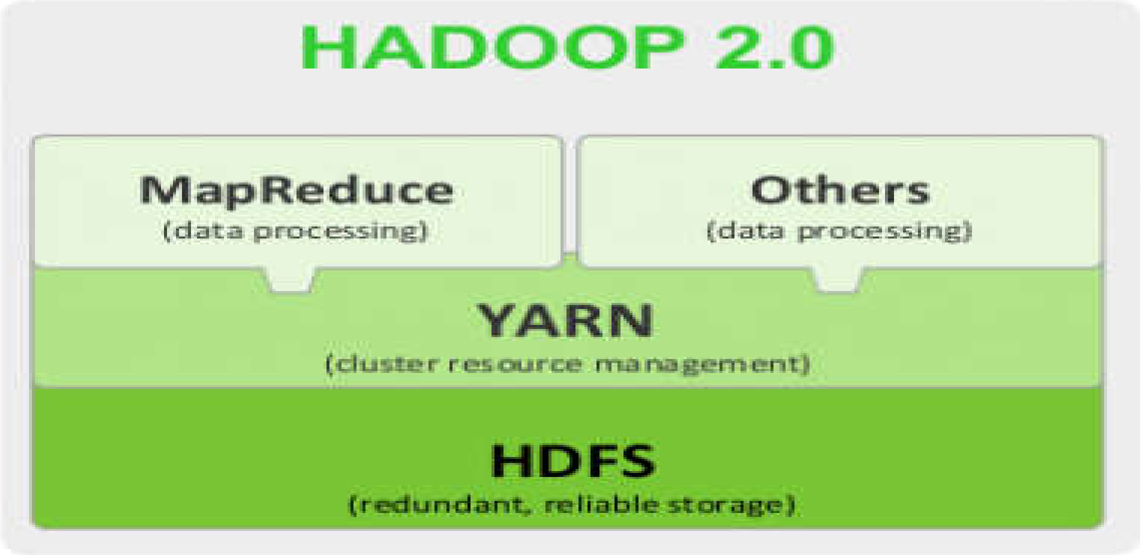
# Wrap up HDFS :

https://www.slideshare.net/eakasit_dpu/introduction-to-hadoop-and-mapreduce-75323926?qid=6dc387d1-f614-4646-a2cb-b4805805235a&v=&b=&from_search=3

# What is MapReduce :

Google released a paper on MapReduce technology in December, 2004. This became the genesis of the Hadoop Processing Model. So, MapReduce is a programming model that allows us to perform parallel and distributed processing on huge data sets.

Google



Principle of MapReduce :

# MapReduce best fit for ? :

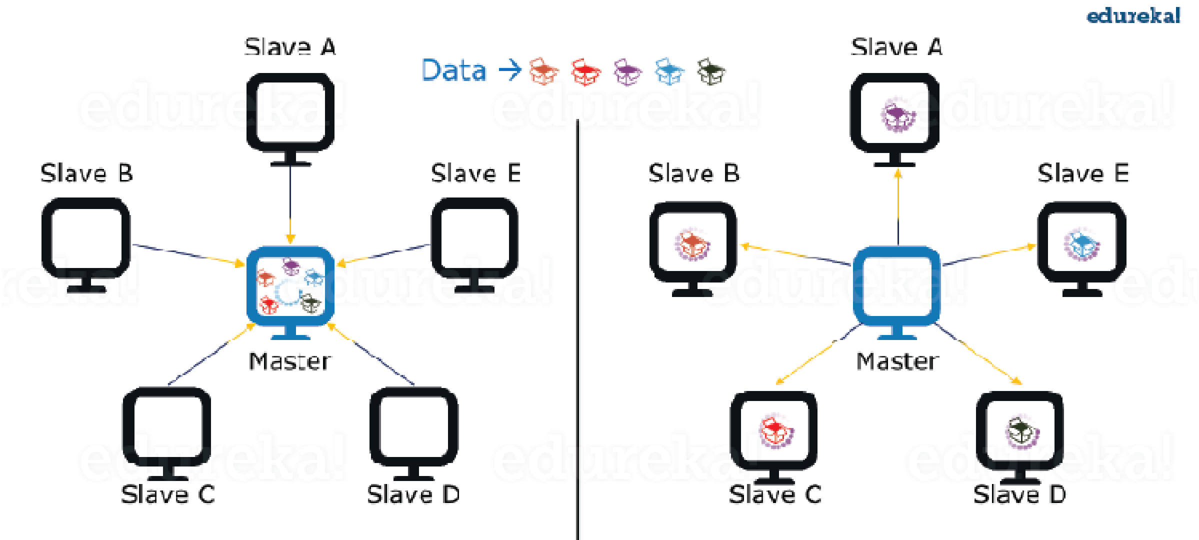
-

# Data Processing Model : Traditional vs. Distributed

The biggest advantage of MapReduce :

1. Parallel Processing:

2. Data Locality:

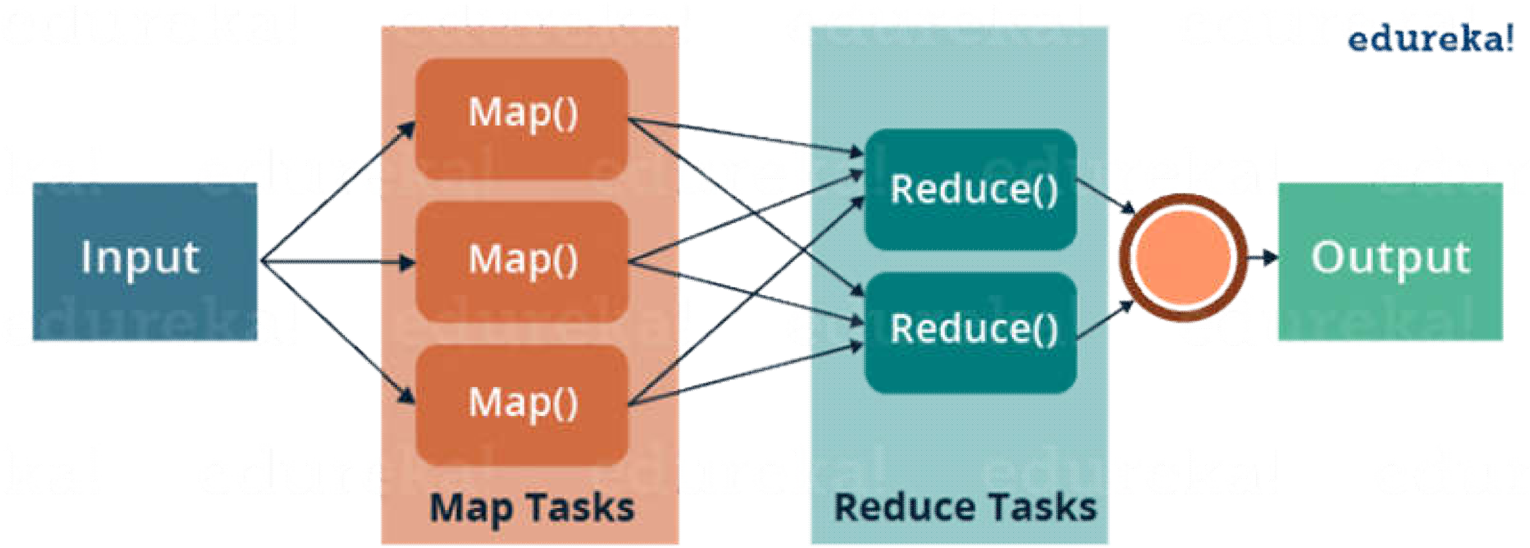


1. Moving data to the Processing Unit (Traditional Approach)

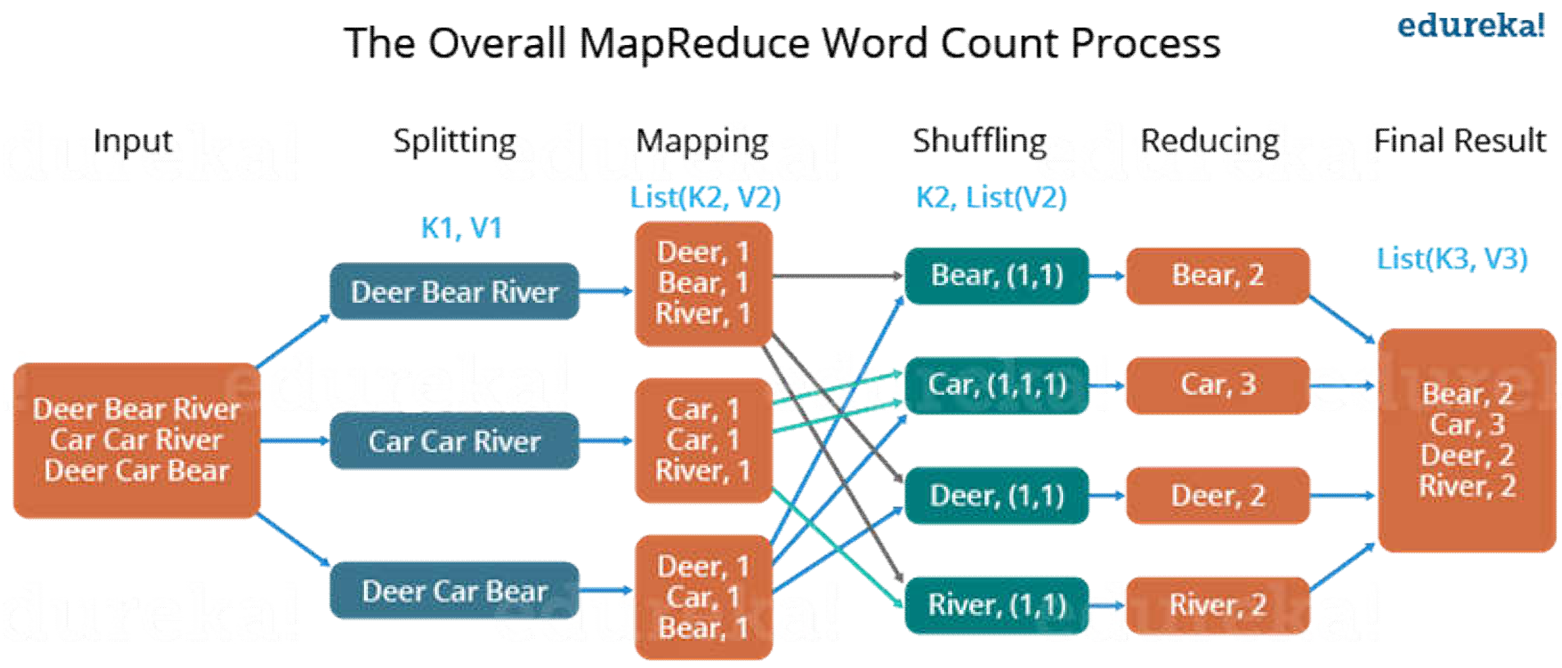
2. Moving Processing Unit to the data (MapReduce Approach)

# MapReduce Concept :

What is MapReduce?

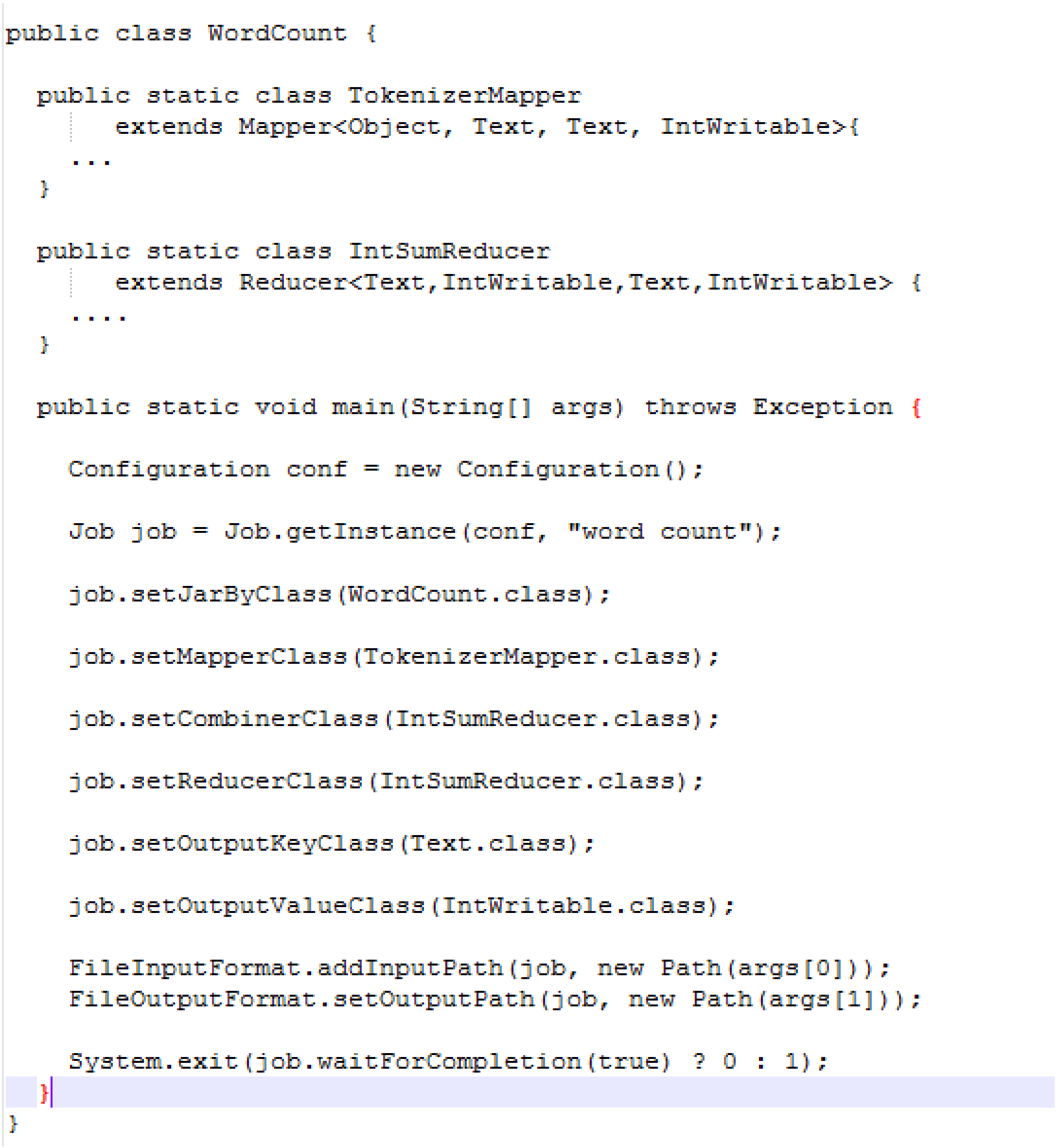


# MapReduce Process Flow :



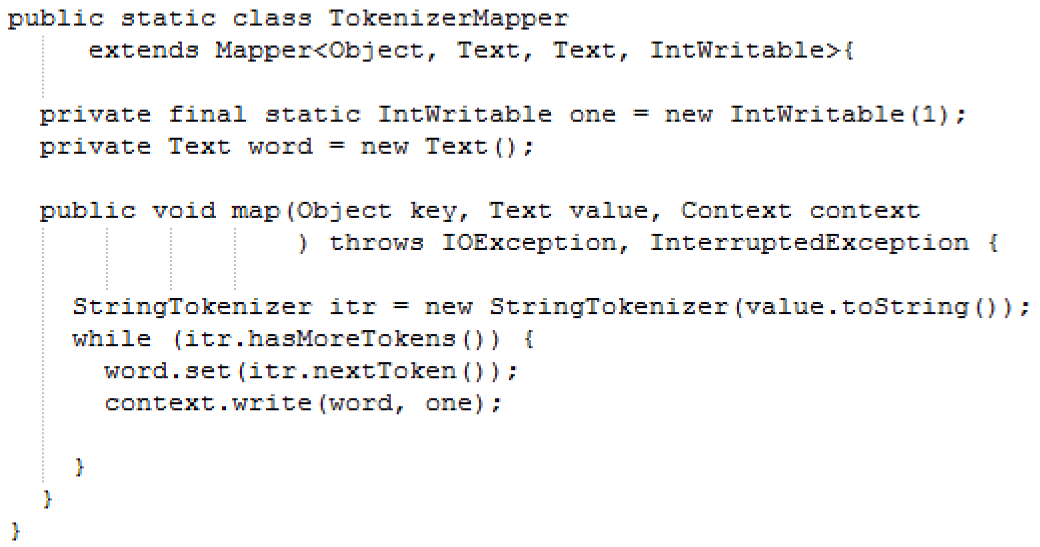
https://www.edureka.co/blog/anatomy-of-a-mapreduce-job-in-apache-hadoop/?utm_source=blog&utm_medium=bottom-related-blog-blogs&utm_campaign=mapreduce-tutorial

# MapReduce Programming Model :

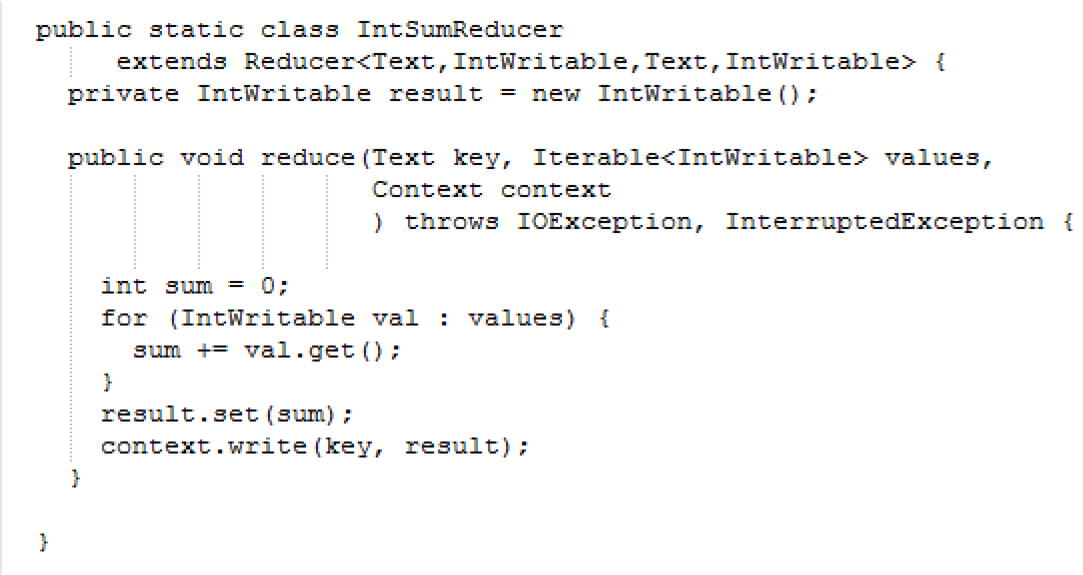


```
public class WordCount {

  public static class TokenizerMapper
       extends Mapper<Object, Text, Text, IntWritable>{
    ...
  }

  public static class IntSumReducer
       extends Reducer<Text,IntWritable,Text,IntWritable> {
    ....
  }

  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);
  }
}
```

# MapReduce Programming Model :

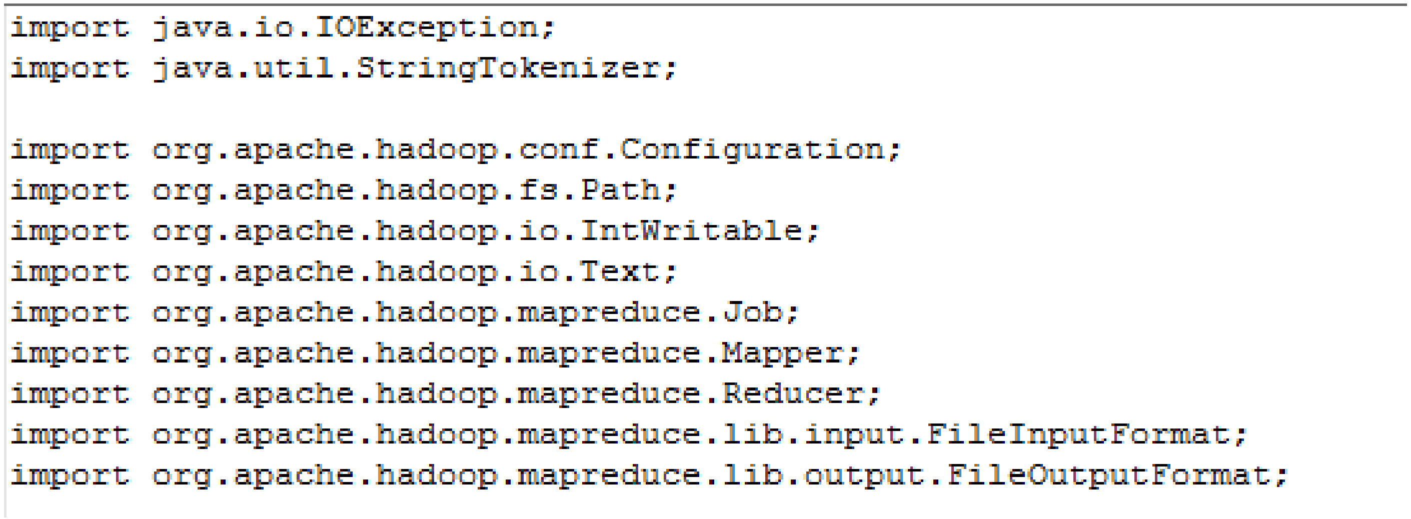


```
public static class TokenizerMapper
     extends Mapper<Object, Text, Text, IntWritable>{

  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);

    }
  }
}
```



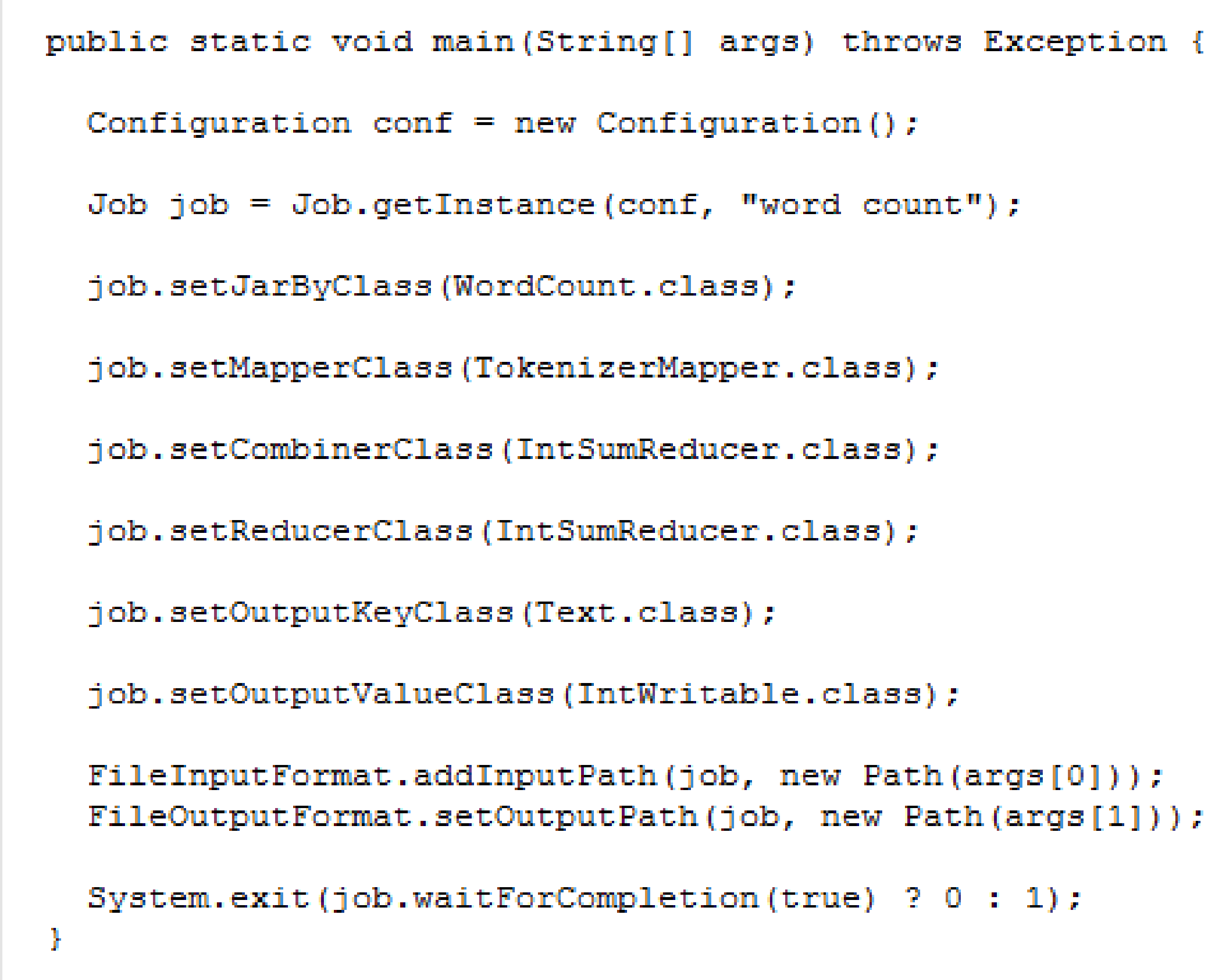
```
public static class IntSumReducer
     extends Reducer<Text,IntWritable,Text,IntWritable> {
  private IntWritable result = new IntWritable();

  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);
  }

}
```

# MapReduce Programming Model :



```
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;
```

MapReduce Programming Model :



```
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);
}
```

# MapReduce Programming :

1). Write Code Program

1-1: Create Directory /home/auoychai/units

/home/auoychai/<xxxx>.java

2). Compile & Package

```
$javac –classpath hadoop-core-1.2.1.jar -d units  <xxxx>.java
$jar –cvf  <xxxx>.jar –C units/ .
```

3). Run Job-Map/Reduce

# Hand-On : File System

1). Create Directory ใน Hadoop

/user/ [lab | dataset]

/lab/[input | output]

/dataset/[csv | txt | json | db

2). Copy file เข้า Hadoop

/home/auoychai/dataset

** copy file ของแต่ละหมวดหมู่ไปไว้ใน Hadoop Directory /user/dataset/ […] ตามกลุ่มของประเภทไฟล์

3). List Directory ใน Hadoop

4). ลบ File / Directory ใน Hadoop

** Delete file ของแต่ละหมวดหมู่ไปไว้ใน Hadoop Directory /user/dataset/ […]

# Hand-On : File System : MapReduce Programming :

1). Write Code Program

1-1: Create Directory /home/auoychai/units

/home/auoychai/Wordcount.java

2). Compile & Package

```
$javac –classpath hadoop-core-1.2.1.jar -d units WordCount.java
$jar –cvf  WordCount.jar –C units/ .
```

3). Run Job-Map/Reduce

```
$hadoop jar WordCount.jar WordCount /user/lab/input/sample.txt
/user/lab/output/o3
```

Principle HDFS & MapReduce :

Wrap up

The End