

# Hadoop Basics with InfoSphere BigInsights

*Lesson 3: Introduction to MapReduce*



## **An IBM Proof of Technology**

---

Catalog Number

---

## Contents

Lab 1Using MapReduce.....	4
1.1Running a MapReduce program (Terminal).....	5
1.2Running a MapReduce program (Web Console).....	7
1.3Running a MapReduce program (Eclipse).....	12
1.3.1Preperation.....	12
1.3.2Create a BigInsights Project.....	12
1.3.3Creating a new java map/reduce program.....	14
1.3.4Implementing a map/reduce program.....	18
1.3.5Running a Java map/reduce program locally.....	19
1.3.6Running a Java map/reduce program on the cluster.....	23
1.4Summary.....	26

---

## Lab 1 Using MapReduce

Now that we've seen how the FileSystem (fs) shell can be used to execute Hadoop commands to interact with HDFS, the same fs shell can be used to launch MapReduce jobs. In this section, we will walk through the steps required to run a MapReduce program. The source code for a MapReduce program is contained in a compiled .jar file. Hadoop will load the JAR into HDFS and distribute it to the data nodes, where the individual tasks of the MapReduce job will be executed. Hadoop ships with some example MapReduce programs to run. One of these is a distributed WordCount program that reads text files and counts the frequency of each word.

After completing this hands-on lab, you'll be able to:

- Run MapReduce programs through terminal
- Run MapReduce programs through Web Console
- Run MapReduce programs through Eclipse.

Allow 45 minutes to 1 hour complete this section of lab.

This version of the lab was designed using the InfoSphere BigInsights 2.1 Quick Start Edition. Throughout this lab you will be using the following account login information:

	Username	Password
VM image setup screen	root	password
Linux	biadmin	biadmin

If you are continuing this series of hands on labs immediately after completing Hadoop Basics Part 1: Exploring Hadoop Distributed File System, you may move on to section 1.1 of this lab. Otherwise please refer to Hadoop Basics Unit 1: Exploring Hadoop Distributed File System Section 1.1 to get started. (All Hadoop components should be started)

## 1.1 Running a MapReduce program (Terminal)

First we need to copy the data files from the local file system to HDFS.

- \_\_1. Open a Terminal and enter `cd $BIGINSIGHTS_HOME/bin`



- \_\_2. Execute the commands below to copy the input files into HDFS.

```
hadoop fs -mkdir /user/biadmin/input
```

```
hadoop fs -put /home/biadmin/sampleData/IBMWatson/*.txt /user/biadmin/input
```

```

biadmin@bivm:~
File Edit View Terminal Tabs Help
[biadmin@bivm ~]$ hadoop fs -mkdir /user/biadmin/input
[biadmin@bivm ~]$ hadoop fs -put /home/biadmin/sampleData/IBMWatson/*.txt /user/biadmin/input
[biadmin@bivm ~]$

```

- \_\_3. Review the files have been copied with the following command:

```
hadoop fs -ls input
```

```

biadmin@bivm:~
File Edit View Terminal Tabs Help
[biadmin@bivm ~]$ hadoop fs -ls input
Found 2 items
-rw-r--r--  1 biadmin supergroup    1416881 2013-06-13 15:21 /user/biadmin/input/blogs-data.txt
-rw-r--r--  1 biadmin supergroup    2634155 2013-06-13 15:21 /user/biadmin/input/news-data.txt
[biadmin@bivm ~]$

```

- \_\_4. Now we can run the wordcount job with the command below.

```
hadoop jar /opt/ibm/biginsights/IHC/hadoop-examples-1.1.1.jar wordcount /user/biadmin/input output
```

*/user/biadmin/input/* is where the input files are, and *output* is the directory where the output of the job will be stored. The *output* directory will be created automatically when executing the command above.

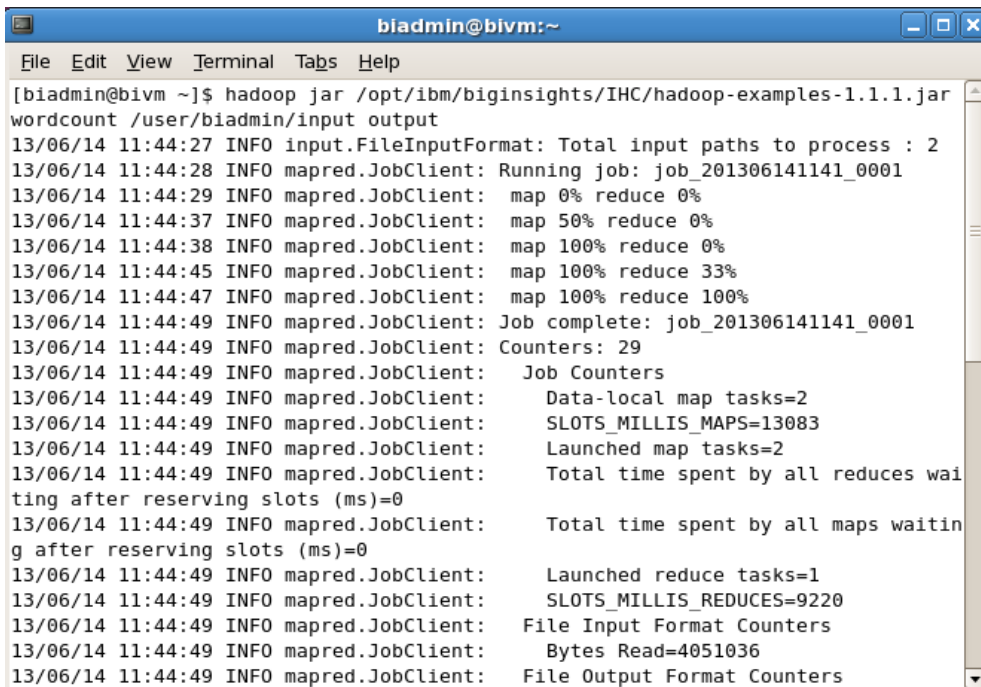


**NOTE:** If the output folder already exists or if you try to rerun a successful Mapper job with the same parameters, the default behavior of MapReduce is to abort the processing. You should see an output error like this:

```

13/05/16 08:38:41 ERROR security.UserGroupInformation:
PrivilegedActionException as:biadmin
cause:org.apache.hadoop.mapred.FileAlreadyExistsException:
Output directory output already exists

```



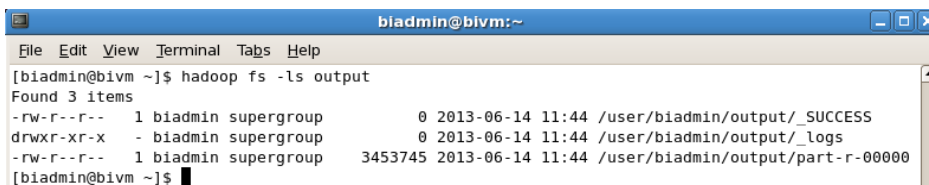
```

biadmin@bivm:~$ hadoop jar /opt/ibm/biginsights/IHC/hadoop-examples-1.1.1.jar
wordcount /user/biadmin/input output
13/06/14 11:44:27 INFO input.FileInputFormat: Total input paths to process : 2
13/06/14 11:44:28 INFO mapred.JobClient: Running job: job_201306141141_0001
13/06/14 11:44:29 INFO mapred.JobClient: map 0% reduce 0%
13/06/14 11:44:37 INFO mapred.JobClient: map 50% reduce 0%
13/06/14 11:44:38 INFO mapred.JobClient: map 100% reduce 0%
13/06/14 11:44:45 INFO mapred.JobClient: map 100% reduce 33%
13/06/14 11:44:47 INFO mapred.JobClient: map 100% reduce 100%
13/06/14 11:44:49 INFO mapred.JobClient: Job complete: job_201306141141_0001
13/06/14 11:44:49 INFO mapred.JobClient: Counters: 29
13/06/14 11:44:49 INFO mapred.JobClient: Job Counters
13/06/14 11:44:49 INFO mapred.JobClient: Data-local map tasks=2
13/06/14 11:44:49 INFO mapred.JobClient: SLOTS_MILLIS_MAPS=13083
13/06/14 11:44:49 INFO mapred.JobClient: Launched map tasks=2
13/06/14 11:44:49 INFO mapred.JobClient: Total time spent by all reduces wait
ting after reserving slots (ms)=0
13/06/14 11:44:49 INFO mapred.JobClient: Total time spent by all maps waitin
g after reserving slots (ms)=0
13/06/14 11:44:49 INFO mapred.JobClient: Launched reduce tasks=1
13/06/14 11:44:49 INFO mapred.JobClient: SLOTS_MILLIS_REDUCE=9220
13/06/14 11:44:49 INFO mapred.JobClient: File Input Format Counters
13/06/14 11:44:49 INFO mapred.JobClient: Bytes Read=4051036
13/06/14 11:44:49 INFO mapred.JobClient: File Output Format Counters

```

\_\_5. Now review the output of step 3.

```
hadoop fs -ls output
```



```

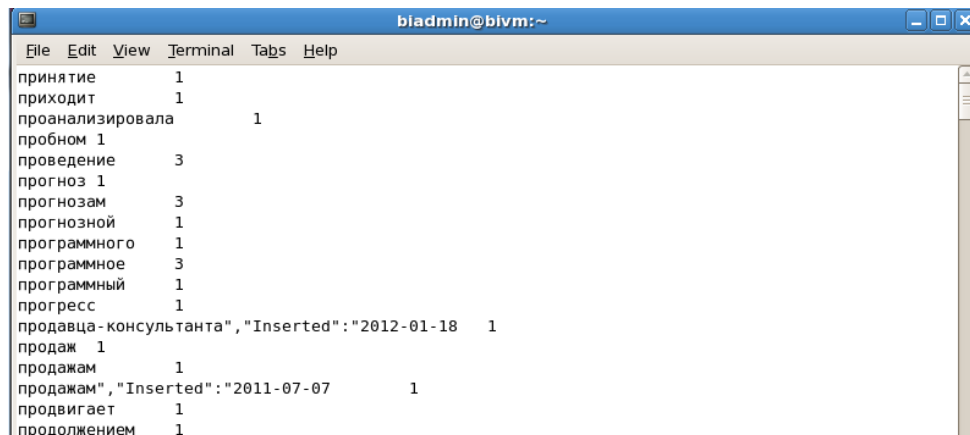
biadmin@bivm:~$ hadoop fs -ls output
Found 3 items
-rw-r--r-- 1 biadmin supergroup          0 2013-06-14 11:44 /user/biadmin/output/_SUCCESS
drwxr-xr-x - biadmin supergroup          0 2013-06-14 11:44 /user/biadmin/output/_logs
-rw-r--r-- 1 biadmin supergroup 3453745 2013-06-14 11:44 /user/biadmin/output/part-r-00000
biadmin@bivm:~$

```

In this case, the output was not split into multiple files (i.e. part-r-00001, part-r-00002, etc)

\_\_6. To view the contents of *part-r-0000* file issue the command below:

```
hadoop fs -cat output/*00
```



```

биadmin@bivm:~$ hadoop fs -cat output/*00
принятие 1
приходит 1
проанализировала 1
пробном 1
проведение 3
прогноз 1
прогнозам 3
прогнозной 1
программного 1
программное 3
программный 1
прогресс 1
продавца-консультанта", "Inserted": "2012-01-18 1
продаж 1
продажам 1
продажам", "Inserted": "2011-07-07 1
продвигает 1
продолжением 1

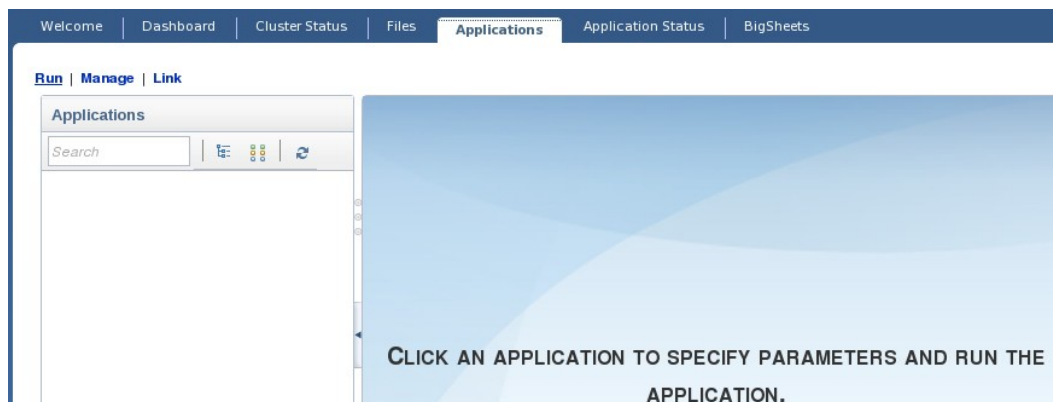
```

## 1.2 Running a MapReduce program (Web Console)

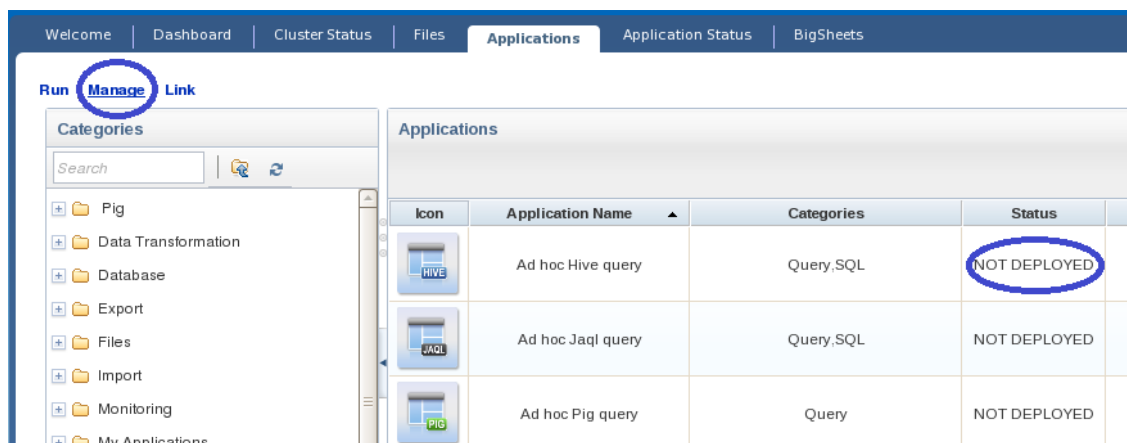
- \_\_1. Launch the Web Console by clicking on the BigInsights WebConsole icon



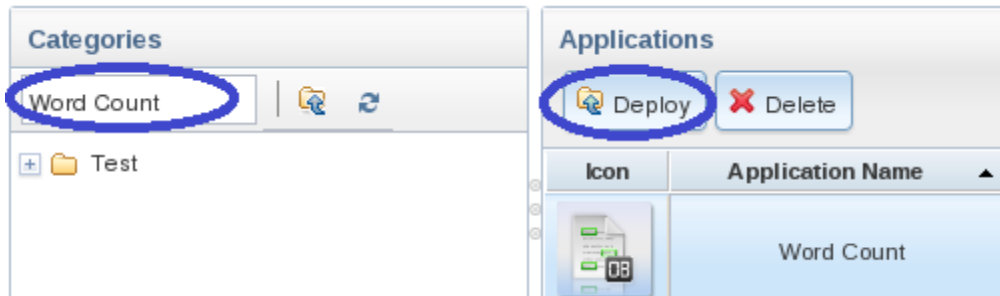
- \_\_2. Click on the **Applications** tab. An application catalog is displayed in the pane at left. As you can see there are currently no deployed applications on this VM. (Administrators can upload and “publish” in-house or third-party applications as desired, as well as remove sample applications).



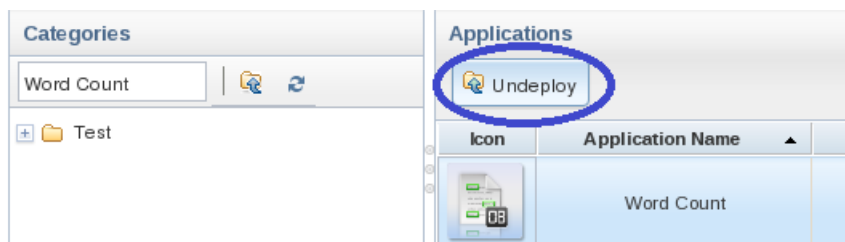
- \_\_3. Click on the **Manage** link. The left pane is a tree of folders that contain sample applications. The right pane is a list of the actual sample applications. Notice how they all say not deployed.



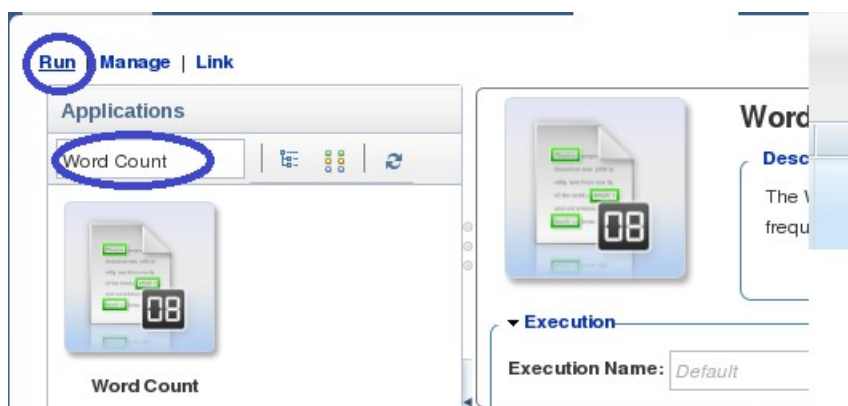
- \_\_\_4. In the search box, type in **Word Count** to search for the word count applications. Once you see the application in the right pane select it. The Web Console displays information about this applications and status in the pane at right. If it is not yet deployed, click on the **Deploy** button as shown below, then click Deploy on the modal dialog that pops up.



- \_\_\_5. When the operation completes, verify that the Web console indicates that the application was successfully deployed. Specifically, confirm that the button in the upper right corner has changed to **Undeploy** and that the **Delete** button beside it is no longer active. (Deployed applications can't be deleted; they must be "undeployed" first, and then they can be deleted from the catalog.) You're now ready to run this application.

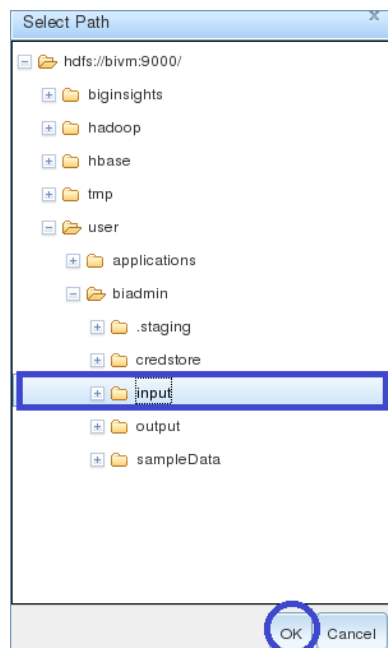


- \_\_\_6. Click on the **Run** link to view deployed applications including the Word Count application you just deployed. If desired, you can begin to type "Word Count" into the search box to quickly access it when you are in the Run pane.





- \_\_\_7. Once you have selected the Word Count Application shown above, in the **Execution Name** box, enter **MapReduceTest**. Do NOT click the **Run** button yet.
- \_\_\_8. In the **Parameters** area, specify the **Input path** directory by clicking on the **Browse** button.
- \_\_\_9. When a pop-up window appears, expand the HDFS directory tree to locate the **/user/biadmin/input** directory. (As you'll recall, you created this directory in a previous module of this lab and uploaded some sample text files to it.) Highlight the directory and click **OK**.





- \_\_\_10. Set the **Output path** parameter of application to the **/user/biadmin/output\_WC** (You can create this directory by browsing for the *output* directory as you did for the *input* directory and then manually appending **\_WC** to it).
- \_\_\_11. Verify that your Word Count settings are consistent with those shown below, and press the green **Run** button. This will cause a simple Oozie-based workflow to be generated for your job (or application), and the application will begin to run.



- \_\_\_12. While the application is running, monitor its progress in the bottom right pane (titled **Application History**). Note that you can stop the application by clicking on the red Stop button in this pane. However, for this exercise, allow the job to run to completion. Depending on your machine resources, this may take a few minutes.

	MapReduceTest	<div><div></div></div> 0%	2	N/A	
---	---------------	---------------------------	---	-----	---

- \_\_\_13. Many applications, including **Word Count**, generate output when successfully executed. In the **Application History** pane (where you monitored the execution status of your **MapReduceTest** application), click on the file pointer in the **Output** column of your job. If necessary use the bottom scroll bar.

Execution Name	Progress	Elapsed Time (sec)	Output	Details
No filter applied				
MapReduceTest	<div><div></div></div> 100%	41		

- \_\_\_14. Your console will switch to the **Files** view and display the directory you specified as the output path for your application (**/user/biadmin/output\_WC** in our example). Expand the contents of this directory until you reach the output file, which is named **part-r-00000**. (If necessary, click on your Web browser's Refresh button.) Click on this file, and the right pane will display a portion of its contents in text format.

Welcome | Dashboard | Cluster Status | **Files** | Applications | Application Status | BigSheets

HDFS

hbase

tmp

user

applications

biadmin

.staging

credstore

input

oozie-biadmin

output

output\_WC

\_SUCCESS

\_logs

part-r-00000

Path: /user/biadmin/output\_WC/part-r-00000

Name	Size	Block
part-r-00000	3.3 MB	128

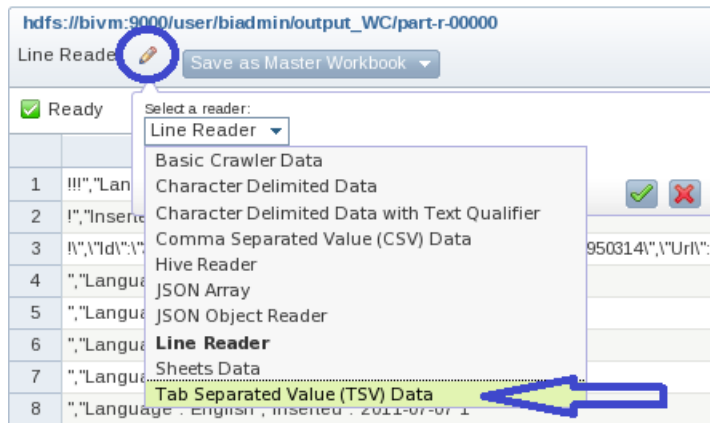
Viewing Size: 10KB | Text | Sheets

!!!, "Language": "English", "Inserted": "2011-02-13  
!", "Inserted": "2012-03-27 1  
!\", \"Id\": \"36034049\", \"ExtKey\": \"7d5743e76c1b075d:  
\": \"http://cestpasmonidee.blogspot.com/\"}, \"Publish  
\", \"Language\": \"Chinese 1  
\", \"Language\": \"English\", \"Inserted\": \"2011-02-17 1  
\", \"Language\": \"English\", \"Inserted\": \"2011-06-20 1  
\", \"Language\": \"English\", \"Inserted\": \"2011-06-24 1  
\", \"Language\": \"English\", \"Inserted\": \"2011-07-07 1  
\", \"Language\": \"English\", \"Inserted\": \"2011-07-08 1  
\", \"Language\": \"English\", \"Inserted\": \"2011-07-10 1  
\", \"Language\": \"English\", \"Inserted\": \"2011-07-29 1  
\", \"Language\": \"English\", \"Inserted\": \"2011-08-02 2  
\", \"Language\": \"English\", \"Inserted\": \"2011-08-05 1  
\", \"Language\": \"English\", \"Inserted\": \"2011-08-09 1  
\", \"Language\": \"English\", \"Inserted\": \"2011-08-16 1

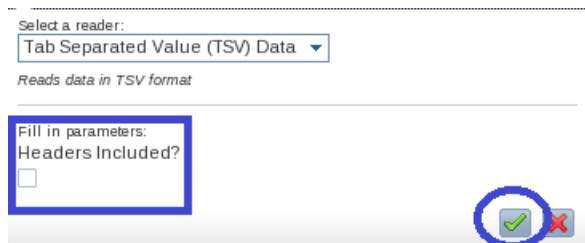
- \_\_15. Because **Word Count** generates output in a tab-separated values format, you can easily view this data using BigInsight's spreadsheet-like tool. In the right pane, change the display style from **Text** to **Sheet** by clicking on the **Sheet** button.



- \_\_16. We want to clearly see this data, in order to do so we need to change the format representation. Click on the pencil icon near the top, next to Line Reader. Use the drop-down menu to specify **Tab-Separated Value Data** as your reader type. (Line Reader is the default reader specified).



- \_\_17. Uncheck the *Headers included?* box and click the green check mark.



- \_\_18. Verify that your output appears in a spreadsheet-style format.

	header1	header2
1	!!!,"Language": "English", "Inserted": "2011-02-13	1
2	!, "Inserted": "2012-03-27	1
3	!\,"Id": "\36034049", "ExtKey": "\7d5743e76c1b075d7740	0
4	, "Language": "Chinese	1
5	, "Language": "English", "Inserted": "2011-02-17	1
6	, "Language": "English", "Inserted": "2011-06-20	1
7	, "Language": "English", "Inserted": "2011-06-24	1

## 1.3 Running a MapReduce program (Eclipse)

For this section you will be using a csv file called RDBMS\_data.csv. This file should be in your local file system under the /home/biadmin/sampleData/DBMS directory.

### 1.3.1 Preparation

You will need to put a copy of sampleData into HDFS.

\_\_19. Launch a terminal and enter the following command:

```
hadoop fs -copyFromLocal /home/biadmin/sampleData
/user/biadmin/sampleData
```

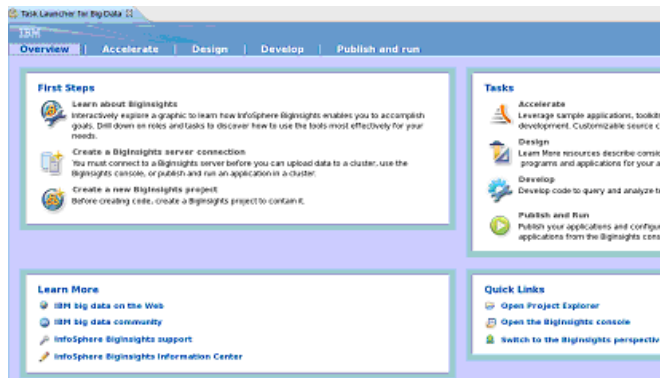
### 1.3.2 Create a BigInsights Project

\_\_20. From your desktop, launch the Eclipse by double-clicking the Eclipse icon. The Eclipse IDE has been pre-loaded with BigInsights tooling. Select the default workspace.



If prompted for a password enter: **biadmin**

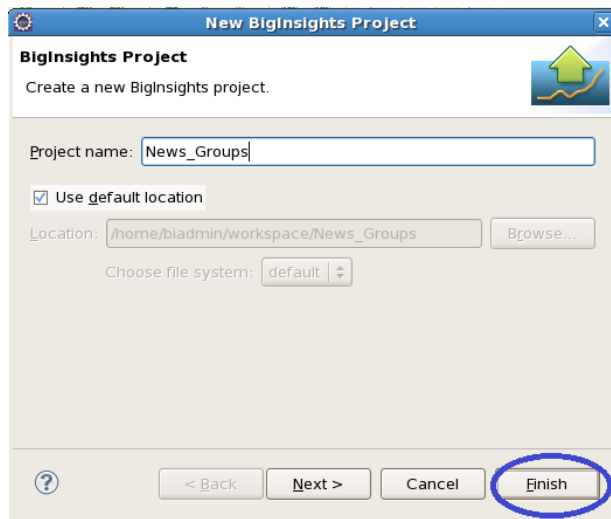
\_\_21. You will be greeted with the Welcome page. It should look similar to the image below.



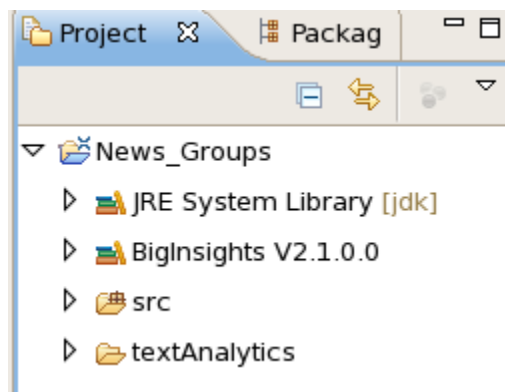
\_\_22. To create a new project, click on 'Create a new BigInsights project' from the First Steps pane in the Overview Tab.



- \_\_23. A prompt asking for project name will appear enter **News\_Groups** and click **Finish**



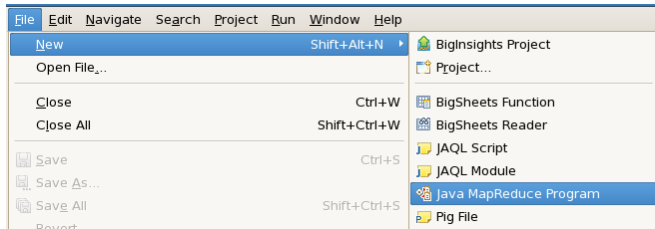
- \_\_24. The project **News\_Groups** will appear in the Project Explorer view. Click the arrow next to **News\_Groups** to expand the project. You will see a **src** folder where all your Java source files will be stored. You'll also see two libraries entries – JRE System Library points to Java to compile your project files, and BigInsights v2.1.0.0 includes all the BigInsights libraries that you will need for development against the BigInsights cluster. This includes Hadoop libraries that are needed to compile Java map/reduce programs. If you use a different version of BigInsights, the BigInsights library entry will show a different version other than 2.1.0.0 and the libraries underneath will have different versions based on what is bundled with the specific version of BigInsights platform.



### 1.3.3 Creating a new java map/reduce program

A Java map/reduce program consists of a mapper, reducer and a driver class that provides the configuration and runs the program. While mapper, reducer and driver methods can all be stored in one class that extends the required interfaces; it is good design to keep them in separate classes. The BigInsights tools for Eclipse provide a wizard that creates templates for all three classes.

- \_\_25. In the **BigInsights** perspective, select **File -> New -> Java MapReduce** program. If the Java MapReduce menu is not visible, you can also select **File -> New -> Other...** and select **Java MapReduce program** from the **BigInsights** folder.

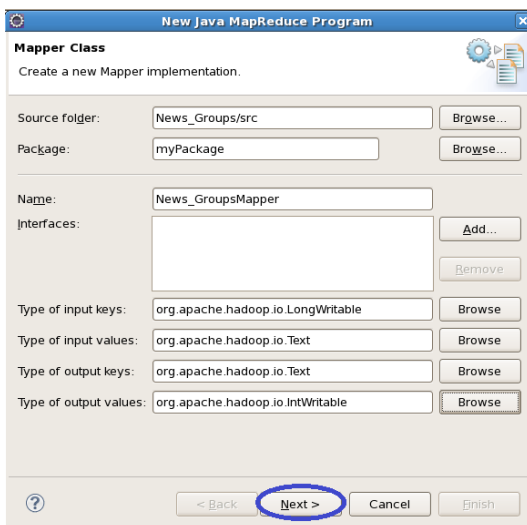


- \_\_26. A new wizard opens which has three pages with the details for a mapper and reducer implementation as well as a driver class.  
For both mapper and reducer we have to specify the input and output key and value pairs.  
First select a package for the mapper class: **myPackage**  
Set the name for the class: **News\_GroupsMapper**

For the sample program, the input will be a csv file and the mapper will be called with one row of data. To be able to check whether we have data, we specify the **input key type** as `org.apache.hadoop.io.LongWritable`. The value is one row of text, so the **input value type** is `org.apache.hadoop.io.Text`.

The mapper will look at the company name, which will be the key for output of the mapper (`org.apache.hadoop.io.Text`) and as the output value we want to get the frequency the company was contacted. (`org.apache.hadoop.io.IntWritable`).

Your first page of the wizard should look like the image below. Click **Next**.



- \_\_27. On the next page, we have to specify the values for the reducer implementation. The package is pre-filled in with the same package as the mapper, but you can change that value if you'd like. Set the name of the reducer class to **News\_GroupsReducer**. The types for the input keys and values are copied from the output keys and values from the mapper class definition because the output of the mapper is processed as input by the reducer.

For the sample program, the reducer is aggregating the number of times each news group was contacted. Hence we set the type of the output keys to Text (org.apache.hadoop.io.Text) and the type of the output values to an integer (org.apache.hadoop.io.IntWritable ).

The wizard page for the reducer should look like the image below. Click **Next**.

**New Java MapReduce Program**

**Reducer Class**  
Create a new Reducer implementation.

Source folder: News\_Groups/src

Package: myPackage

Name: News\_GroupsReducer

Interfaces:

Type of input keys: org.apache.hadoop.io.Text

Type of input values: org.apache.hadoop.io.IntWritable

Type of output keys: org.apache.hadoop.io.Text

Type of output values: org.apache.hadoop.io.IntWritable

- \_\_28. On the last page of the wizard we only have to select a package and specify the name of the driver class. Set **myPackage** as the package name and **News\_GroupsDriver** as the class name. Click **Finish**.

**New Java MapReduce Program**

**Java MapReduce Main class**  
Create a new Java MapReduce Main class.

Source folder: News\_Groups/src

Package: myPackage

Name: News\_GroupsDriver

Interfaces:

After clicking Finish, the wizard will create templates for the three classes and open them in a Java editor. The mapper and reducer classes implement the Hadoop Mapper and Reducer interfaces and it's up to us to implement the methods.

**Mapper:**

```
package myPackage;

import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

public class News_GroupsMapper extends Mapper<LongWritable, Text, Text,
IntWritable> {

    @Override
    public void map(LongWritable key, Text value, Context context)
        throws IOException, InterruptedException {

    }

}
```

**Reducer:**

```
package myPackage;

import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

public class News_GroupsReducer extends Reducer<Text, IntWritable,
Text, IntWritable> {

    public void reduce(Text key, Iterable<IntWritable> values, Context
context)

        throws IOException, InterruptedException {

    }

}
```

The driver class creates a new Hadoop job configuration and sets mapper and reducer classes accordingly.

```
package myPackage;

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.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.util.GenericOptionsParser;
public class News_GroupsDriver {
    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        // Use programArgs array to retrieve program arguments.
        String[] programArgs = new GenericOptionsParser(conf, args)
            .getRemainingArgs();
        Job job = new Job(conf);
        job.setJarByClass(SalesDriver.class);
        job.setMapperClass(SalesMapper.class);
        job.setReducerClass(SalesReducer.class);

        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);
        // TODO: Update the input path for the location of the
inputs of the map-reduce job.
        FileInputFormat.addInputPath(job, new Path("[input path]"
);
        // TODO: Update the output path for the output directory of
the map-reduce job.
        FileOutputFormat.setOutputPath(job, new Path("[output path
]"));
        // Submit the job and wait for it to finish.
        job.waitForCompletion(true);
        // Submit and return immediately:
        // job.submit();
    }
}
```

### 1.3.4 Implementing a map/reduce program

Now that we have created the templates for the map/reduce program, we can use the Java editor to implement the mapper and reducer classes and make a few changes to the driver class. Since the tutorial focuses on the tooling aspect and not on the details of implementing a mapper and reducer, we'll just provide the implementation for the relevant methods. Copy the code into the generated templates for each class.

#### News\_GroupsMapper.java:

**map** method

```
public void map(LongWritable key, Text value, Context context)
    throws IOException, InterruptedException {
    // Make sure we have data
    if (key.get() > -1) {
        // Convert each line of data to string for processing
        String line = value.toString();
        // Split up each element of the line
        String[] elements = line.split(",");

        // news group name is second field
        String newsgroup = elements[1];
        context.write(new Text(newsgroup), new IntWritable(1));
    }
}
```

#### News\_GroupsReducer.java:

**reduce** method

```
public void reduce(Text key, Iterable<IntWritable> values, Context
context)
    throws IOException, InterruptedException {
    // Initialize count variable
    int count = 0;
    // Go through each element and count repetitions
    for (IntWritable value : values) {
        count++;
    }
    context.write(key, new IntWritable(count));
}
```

```
}
```

### News\_GroupsDriver.java:

The only change we have to make to the driver class is to get the arguments for the input and output path from the program arguments to be able to specify the values during runtime.

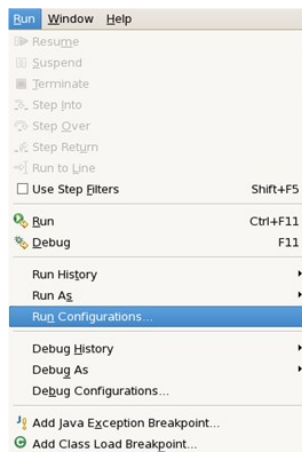
- \_\_29. In the News\_GroupsDriver.java class, go to the sections in the code labeled //TODO and update the code snippets as shown below:

```
FileInputFormat.addInputPath(job, new Path(programArgs[0]));
FileOutputFormat.setOutputPath(job, new Path(programArgs[1]));
```

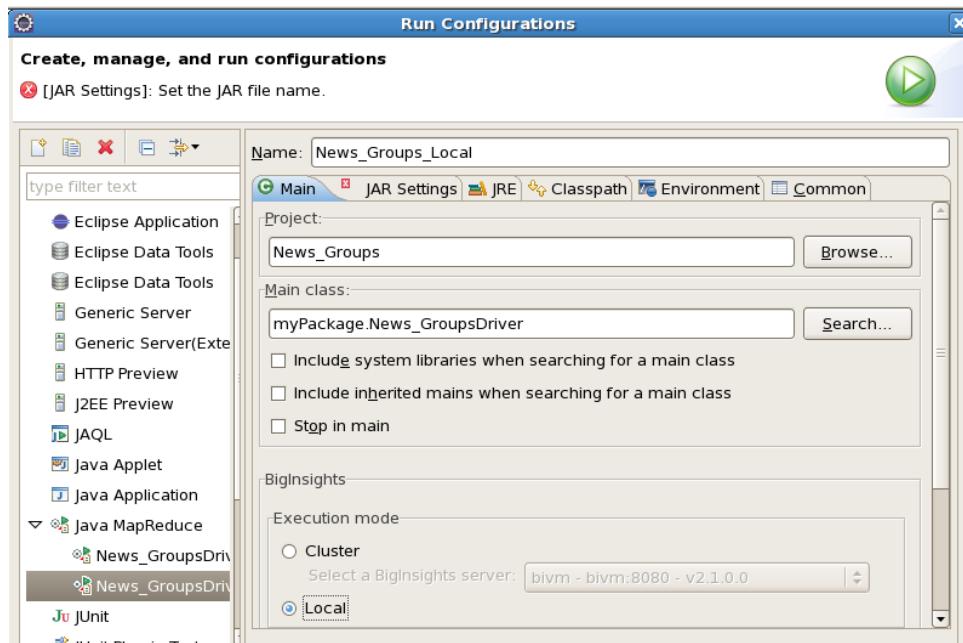
### 1.3.5 Running a Java map/reduce program locally

Now that we have implemented the program, we can run the program. The tooling provides the capabilities to run the Java map-reduce program locally, or it can be run remotely on the cluster. First we will run the program locally.

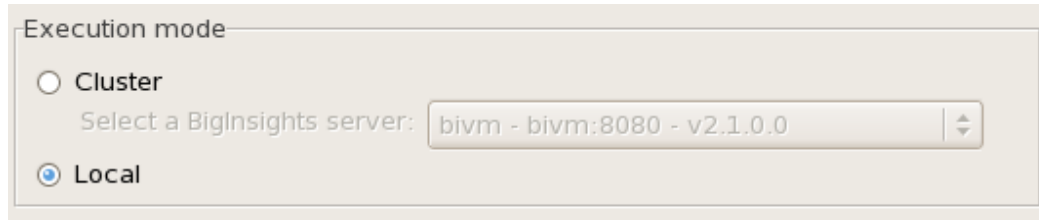
- \_\_30. From the **Run** menu, select **Run Configurations...**



- \_\_31. A new Run configurations dialog opens up. Double-click on the entry **Java MapReduce** in the tree on the left hand side to create a new empty run configuration.
- \_\_32. In the **Main** tab, select the project **News\_Groups**, and select the main class **myPackage.News\_GroupsDriver**. You can also give the run configuration a more meaningful name in the **Name** field.

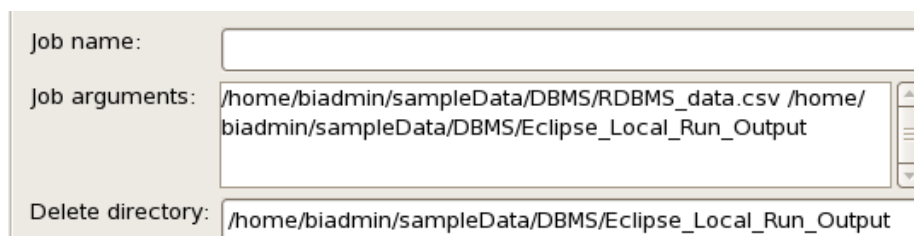


- \_\_33. In the **BigInsights** section, we can select whether we want to run against the cluster (default), or whether we want to run locally. For now we want to run in our local Eclipse machine, so we select **Local**. (scroll down if needed)



- \_\_34. Our program requires two input arguments – a value for the input of the data (directory or a specific file) and a folder where the output will be stored. Since we want to run locally, you should specify directories in your local file system. As input file, point to the location of the file RDBMS\_data.csv in your local file system.  
(/home/biadmin/sampleData/DBMS/RDBMS\_data.csv)

The output directory can be any directory in your local file system where your user id has write access to.

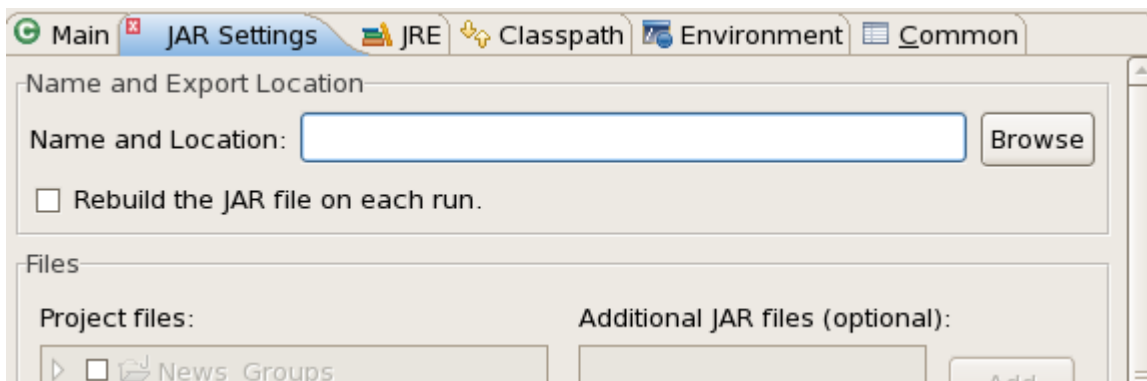


The delete directory is set to the same value as the output directory of the program. If the output directory exists when running the program, Hadoop will throw an error that the directory already exists.

Instead of deleting it manually every time before running the program, we can simply set the delete directory which comes in handy when developing a new program or debugging.

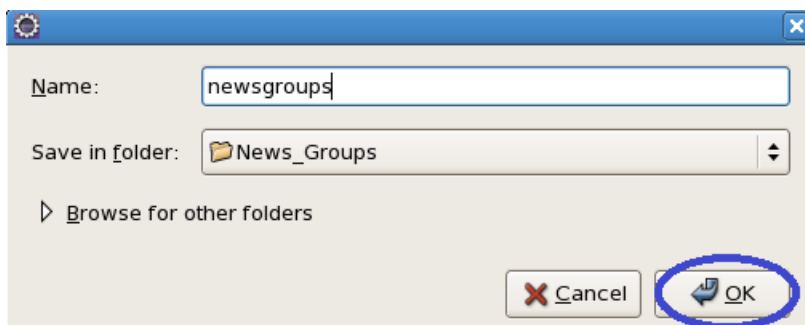
The job name is an optional parameter and you can set it to override the job name that Hadoop will assign automatically when running a job.

- \_\_35. In addition to settings in the Main tab, we'll also have to make a few changes in the **JAR Settings** tab. Click on the **JAR settings** tab to open it

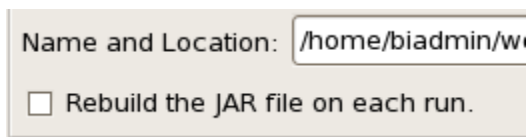


In order to run the map/reduce program, it will be handed to Hadoop as a JAR file. In the JAR settings tab you have the option to specify an existing JAR file or let the JAR rebuild every time you run the program using the source in your BigInsights project.

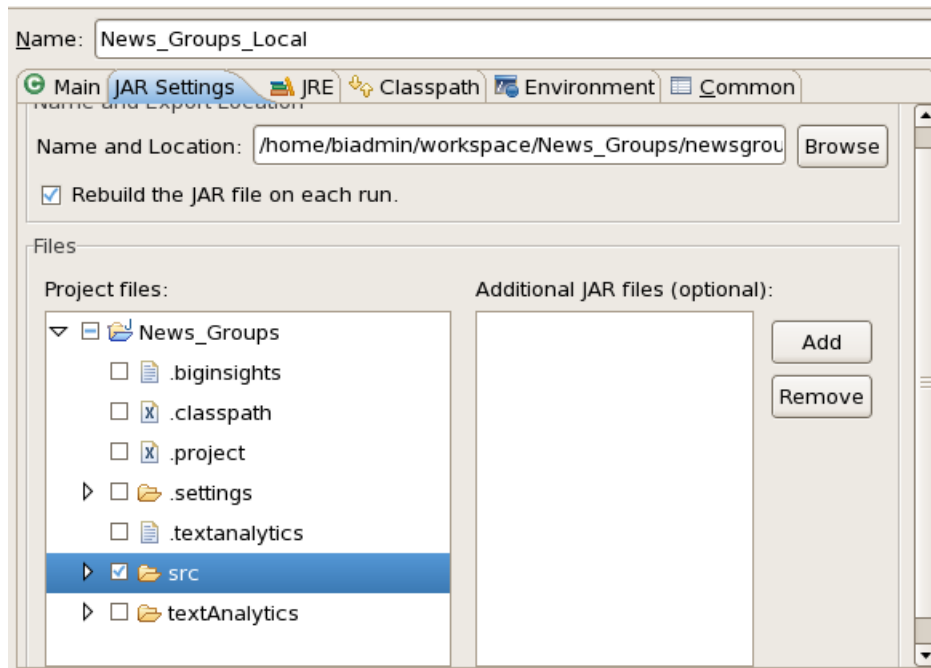
- \_\_36. Click on **Browse** to open a file select dialog, keep the default location (which is the location of your BigInsights project), and type in the name **newsgroups**. Click **OK** to close the Browse dialog.



- \_\_37. In the JAR settings tab, check the checkbox **Rebuild the JAR file on each run**. This will ensure that we rebuild the jar with the latest source files every time you run the program.



- \_\_38. Expand the tree and select the **src** folder to include all Java files. The page should look like the image below and the error indicator next to the JAR settings name should disappear.



In the *Additional JAR files* list, you can add additional JAR files that your program may need and they will be bundled along with the jar and added to the classpath when Hadoop runs your program. Our sample program doesn't require any additional Jar files, so we keep the list empty.

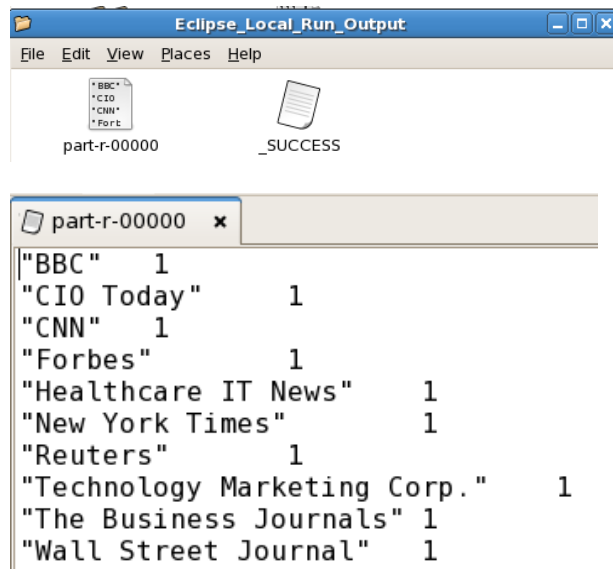
- \_\_39. Click **Apply** and **Run** to run the program. Since we are running locally, you will see a console window with the output from Hadoop.

```
13/06/14 13:31:39 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
13/06/14 13:31:39 INFO input.FileInputFormat: Total input paths to process : 1
13/06/14 13:31:39 WARN snappy.LoadSnappy: Snappy native library not loaded
13/06/14 13:31:39 INFO mapred.JobClient: Running job: job_local_0001
13/06/14 13:31:39 INFO util.ProcessTree: setsid exited with exit code 0
13/06/14 13:31:39 INFO mapred.Task: Using ResourceCalculatorPlugin : org.apache.hadoop.util.LinuxResourceCalculatorPlugin@58c458c4
13/06/14 13:31:39 INFO mapred.MapTask: io.sort.mb = 100
13/06/14 13:31:39 INFO mapred.MapTask: data buffer = 79691776/99614720
13/06/14 13:31:39 INFO mapred.MapTask: record buffer = 262144/327680
13/06/14 13:31:39 INFO mapred.MapTask: Starting flush of map output
13/06/14 13:31:39 INFO mapred.MapTask: Finished spill 0
13/06/14 13:31:39 INFO mapred.Task: Task:attempt_local_0001_m_000000_0 is done. And is in the process of committing
13/06/14 13:31:39 INFO mapred.LocalJobRunner:
13/06/14 13:31:39 INFO mapred.Task: Task 'attempt_local_0001_m_000000_0' done.
13/06/14 13:31:39 INFO mapred.Task: Using ResourceCalculatorPlugin : org.apache.hadoop.util.LinuxResourceCalculatorPlugin@4abd4abd
```

- \_\_40. When the job is completed, you'll see a line that indicates that both map and reduce are 100% done.

```
13/06/14 13:31:39 INFO output.FileOutputCommitter: Saved output of task 'att
13/06/14 13:31:39 INFO mapred.LocalJobRunner: reduce > reduce
13/06/14 13:31:39 INFO mapred.Task: Task 'attempt_local_0001_r_000000_0' don
13/06/14 13:31:40 INFO mapred.JobClient: map 100% reduce 100%
13/06/14 13:31:40 INFO mapred.JobClient: Job complete: job_local_0001
13/06/14 13:31:40 INFO mapred.JobClient: Counters: 20
13/06/14 13:31:40 INFO mapred.JobClient: File Output Format Counters
```

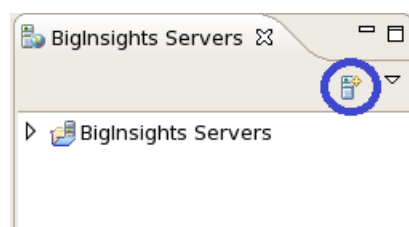
To see the output of the program, navigate to the output directory that you specified in the run configuration using the file browser. You'll see two files: a success file from Hadoop and a part file that contains the output of your program. If you open the part file, you'll see that each news group has been contacted once.



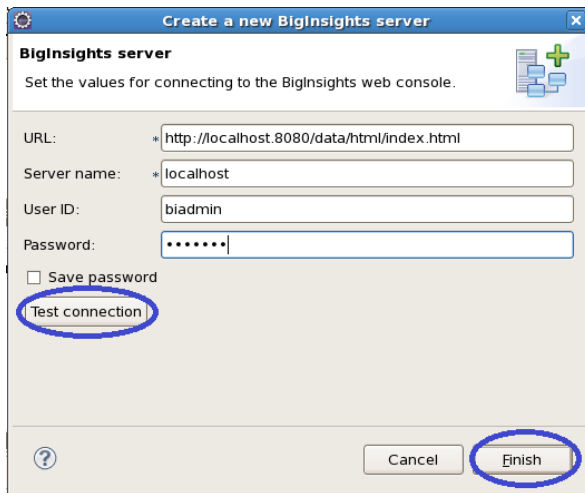
### 1.3.6 Running a Java map/reduce program on the cluster

Now that we have run the program in Eclipse locally, we want to test the program on the cluster as well. To run your program on the cluster, you need to define a connection to an existing BigInsights cluster first.

- \_\_\_41. In the *BigInsights Perspective*, click on the **Add a new server** icon in the *BigInsights servers* view.



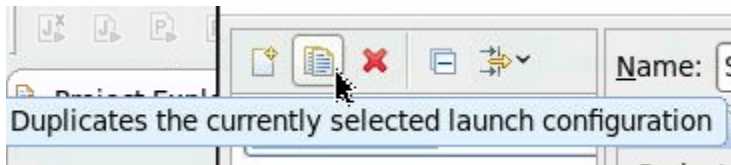
- \_\_\_42. When prompted, specify the URL for your BigInsights web console as well as user id and password. The server name will self-populate once you enter a value into the URL field, but you can change the value.



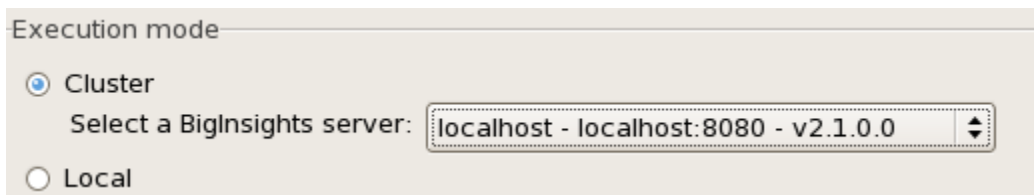
- \_\_43. Click **Test connection** once your connection is successful click **Finish** to register the BigInsights server in the BigInsights Servers view.

To run on the cluster, we still need a run configuration and we can use the one we created for the local run as the base to create a new configuration.

- \_\_44. Click on **Run -> Run Configurations...** and select the previously created run configuration. Because we have to change the locations for input and output directories to reflect the locations in the Hadoop file system, we will create a copy of the run configuration first so that we don't have to fill out everything again and can easily switch between local and cluster mode without changing our program parameters.
- \_\_45. In the run configuration dialog, select the previously created run configuration (News\_Groups\_Local), click the **Duplicate** button to create a copy of the existing run configuration and give the run configuration a new name.



- \_\_46. In the **Execution mode**, select Cluster and select the just registered BigInsights server.



- \_\_47. In the job arguments section, we need to update the input file location to HDFS. Make sure your user id has write access to the output directory in HDFS. Use the same value for the delete directory as for the output directory.

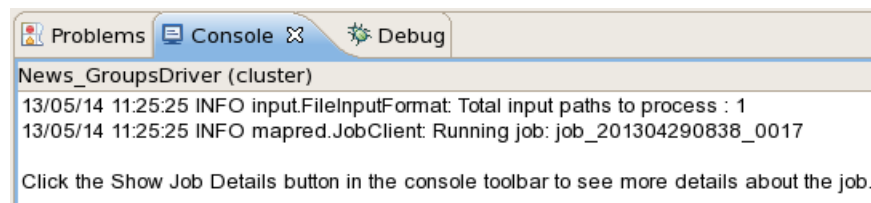


Job name:

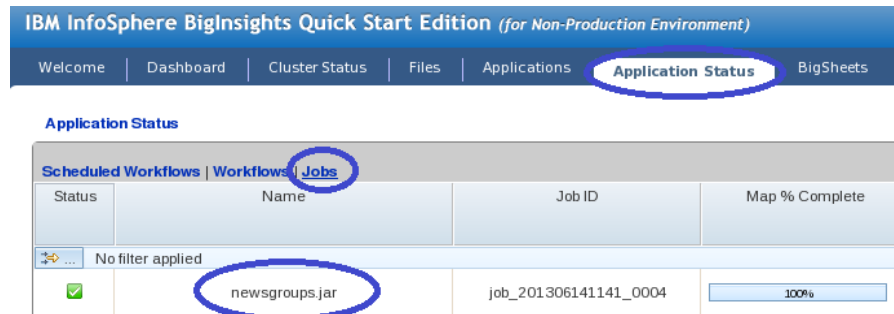
Job arguments:

Delete directory:

- \_\_48. Click Apply and **Run** to run the program on the cluster.
- \_\_49. When the program is run on the cluster, the JAR of the program will be copied to the cluster and run on the cluster using the Hadoop command. A dialog with the progress information will pop up in Eclipse and when the program was successfully submitted on the server, you will see the job id information in the Console window of your Eclipse workspace.



- \_\_50. To see the details of your job, you need to go to the **BigInsights** console. Open the BigInsights Web Console from the desktop.
- \_\_51. Click on the *Application Status* page and click on the sub link *Jobs*.



The jobs page shows the job status information for any Hadoop job and is the starting point to drill down into more job details.

- \_\_52. To drill down into jobs details, click on the row of the job. In the bottom of the page, the breakdown of setup, map, reduce and cleanup tasks is shown in a table and you can see that the map task failed was killed and was not successful.

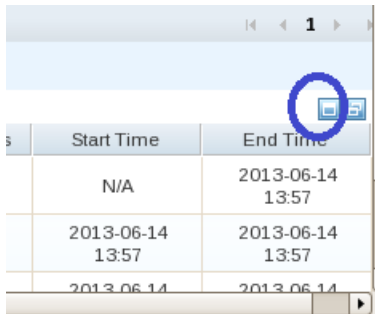
1 - 4 of 4 items 10 | 25 | 50 | 100 | All

Job Counters Job Configuration

All Tasks

Type	Total Tasks	Successful Tasks	Failed Tasks	Killed Tasks	Running Tasks	Pending Tasks	Start Time	End Time
setup	2	1	0	1	0	0	N/A	2013-06-14 13:57
map	1	1	0	0	0	0	2013-06-14 13:57	2013-06-14 13:57

- \_\_\_53. Click on the map row to drill down into more details. Keep clicking on the task attempts until you reach the page with the log files. To have more space to see the log file content, click the Window icon on the right:



In the log file page, you'll see the stdout, stderr and syslog output from the job.

[All Tasks](#) → [Task Details\[map\]](#) → [Task Attempts](#) → [Attempt Log](#)

**stdout logs**

--

**stderr logs**

--

**syslog logs**

```

2013-06-14 13:57:19,060 INFO org.apache.hadoop.util.NativeCodeLoader: Loaded the native
2013-06-14 13:57:19,351 INFO org.apache.hadoop.util.ProcessTree: setsid exited with exit co
2013-06-14 13:57:19,354 INFO org.apache.hadoop.mapred.Task: Using ResourceCalculator
2013-06-14 13:57:19,554 INFO org.apache.hadoop.mapred.MapTask: io.sort.mb = 256
2013-06-14 13:57:20,233 INFO org.apache.hadoop.mapred.MapTask: data buffer = 2040109
2013-06-14 13:57:20,233 INFO org.apache.hadoop.mapred.MapTask: record buffer = 671088
2013-06-14 13:57:20,327 INFO org.apache.hadoop.mapred.MapTask: Starting flush of map o
2013-06-14 13:57:20,336 INFO org.apache.hadoop.mapred.MapTask: Finished spill 0
2013-06-14 13:57:20,342 INFO org.apache.hadoop.mapred.Task: Task:attempt_2013061411
2013-06-14 13:57:20,389 INFO org.apache.hadoop.mapred.Task: Task 'attempt_2013061411
2013-06-14 13:57:20,420 INFO org.apache.hadoop.mapred.TaskLogsTruncater: Initializing lo
2013-06-14 13:57:20,453 INFO org.apache.hadoop.io.nativeio.NativeIO: Initialized cache for l
2013-06-14 13:57:20,453 INFO org.apache.hadoop.io.nativeio.NativeIO: Got UserName biadr

```

## 1.4 Summary

Congratulations! You have now experienced running a java map/reduce program through the Terminal, Web Console, and Eclipse.

## NOTES

[illegible]

## NOTES

[illegible]



© Copyright IBM Corporation 2013.

The information contained in these materials is provided for informational purposes only, and is provided AS IS without warranty of any kind, express or implied. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, these materials. Nothing contained in these materials is intended to, nor shall have the effect of, creating any warranties or representations from IBM or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of IBM software. References in these materials to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates. This information is based on current IBM product plans and strategy, which are subject to change by IBM without notice. Product release dates and/or capabilities referenced in these materials may change at any time at IBM's sole discretion based on market opportunities or other factors, and are not intended to be a commitment to future product or feature availability in any way.

IBM, the IBM logo and [ibm.com](http://ibm.com) are trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at [www.ibm.com/legal/copytrade.shtml](http://www.ibm.com/legal/copytrade.shtml).



Please Recycle