

Java-Success.com

Prepare to fast-track, choose & go places with 800+ Java & Big Data Q&As with lots of code & diagrams.


search here ...

Go

- Home
- Why? ▾
- 300+ Java FAQs ▾
- 300+ Big Data FAQs ▾
- Courses ▾
- Membership ▾
- Your Career ▾

Home > bigdata-success.com > Tutorials - Big Data > TUT - Pig > 1. Apache Pig Getting started

1. Apache Pig Getting started

 Posted on February 1, 2016


Input Data

scores.data in folder: `/Users/arulk/projects`
representing marks of 4 students in 3 subjects:



1	
2	Science, 80, 75, 89, 90
3	Maths, 90, 87, 78, 92
4	English, 78, 88, 65, 99
5	

Calculate the max mark for each subject.

300+ Java Interview FAQs

- 300+ Java FAQs  ▾
- 16+ Java Key Areas Q&As ▾
- 150+ Java Architect FAQs ▾
- 80+ Java Code Quality Q&As ▾
- 150+ Java Coding Q&As ▾

300+ Big Data Interview FAQs

- 300+ Big Data FAQs  ▾
- Tutorials - Big Data >
- TUT -  Starting Big Data
- TUT - Starting Spark & Scala



Step 1: Download Apache Pig from

<http://apache.mirror.digitalpacific.com.au/pig/> and extract the tar file.

```
1
2 tar -zxvf ./downloads/pig-0.15.0.tar.gz
3
```

Step 2: Set up PIG_HOME and add \$PIG_HOME/bin to the path via .profile or .bashrc.

```
1
2 export JAVA_HOME=/Library/Java/JavaVirtualMachines
3 export M3_HOME=~/.tools/apache-maven-3.3.9
4 export HADOOP_HOME=~/.hadoop-eco/hadoop-2.7.1
5 export HADOOP_MAPRED_HOME=$HADOOP_HOME
6 export HADOOP_COMMON_HOME=$HADOOP_HOME
7 export HADOOP_HDFS_HOME=$HADOOP_HOME
8 export YARN_HOME=$HADOOP_HOME
9 export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/
10
11 export HBASE_HOME=~/.hbase-1.0.3
12
13 export PIG_HOME=~/.pig-0.15.0
14
15 export PATH=$PATH:$M3_HOME/bin:$JAVA_HOME/bin:$HAI
16 export HADOOP_INSTALL=$HADOOP_HOME
17
```

```
1
2 $source .profile # or .bashrc
3
```

Step 3: Using the '-x local' options starts pig in the local mode whereas executing the pig command without any options starts in Pig in the cluster mode. When in local mode, pig can access files on the local file system. In cluster mode, pig can access files on HDFS.

```
1
2 $pig -x local
```

TUT - Starting with Python

TUT - Kafka

TUT - Pig

TUT - Apache Storm

TUT - Spark Scala on Zeppelin

TUT - Cloudera

TUT - Cloudera on Docker

TUT - File Formats

TUT - Spark on Docker

TUT - Flume

TUT - Hadoop (HDFS)

TUT - HBase (NoSQL)

TUT - Hive (SQL)

TUT - Hadoop & Spark

TUT - MapReduce

TUT - Spark and Scala

TUT - Spark & Java

TUT - PySpark on Databricks

TUT - Zookeeper

800+ Java Interview Q&As

300+ Core Java Q&As



300+ Enterprise Java Q&As



150+ Java Frameworks Q&As



120+ Companion Tech Q&As



Tutorials - Enterprise Java



3

Step 4: Read the file

“/Users/arulk/projects/scores.data”.

```
1  
2 grunt> student_marks = LOAD '/Users/arulk/project:  
3
```

Note: If data type is not provided, the default data type is “**byte array**”

The “student_marks” is known as a “**relation**”, and NOT a variable. Pig is a **data flow language**. A Pig relation is a bag of **tuples**. A Pig relation is similar to a table in a relational database, where the tuples in the bag correspond to the rows in a table.

A **tuple** is just like a row in a table. It is comma separated list of fields.

```
1  
2 (Science, 80, 75, 89, 90)  
3
```

A **bag** is an unordered collection of tuples.

```
1  
2 {(80.0),(75.0),(89.0),(90.0)}  
3
```

Three handy commands to check the **structure (aka schema)**, how the data flow was derived and the **actual data** are:

1



```

2 grunt> describe student_marks
3 grunt> illustrate student_marks
4 grunt> dump student_marks
5

```

student_marks	Subject:chararray	Student1:double	Student2:double	Student3:double	Student4:double
	Science	80	75	89	90

illustrate student_marks

```

1
2 grunt> dump student_marks
3
4 (Science, 80, 75, 89, 90)
5 (Maths, 90, 87, 78, 92)
6 (English, 78, 88, 65, 99)
7
8

```

Step 5: Bag student mark columns.

```

1
2 grunt> bagged = foreach student_marks generate Subject
3

```

```

1
2 grunt> dump bagged
3
4 (Science,{(80.0),(75.0),(89.0),(90.0)})
5 (Maths,{(90.0),(87.0),(78.0),(92.0)})
6 (English,{(78.0),(88.0),(65.0),(99.0)})
7
8

```

Step 6: Flattening the Bag.

```

1
2 grunt> pivoted_1 = foreach bagged generate Subject
3

```

```

1
2 grunt> dump pivoted_1;
3
4 (Science,80.0)
5 (Science,75.0)
6 (Science,89.0)

```



```

7 (Science,90.0)
8 (Maths,90.0)
9 (Maths,87.0)
10 (Maths,78.0)
11 (Maths,92.0)
12 (English,78.0)
13 (English,88.0)
14 (English,65.0)
15 (English,99.0)
16
17

```

Step 7: Group it by Subject.

```

1
2 grunt> records_group = GROUP pivoted_1 by Subject;
3

```

```

1
2 grunt> dump records_group;
3
4 (Maths,{(Maths,92.0),(Maths,78.0),(Maths,87.0),(Ma
5 (English,{(English,99.0),(English,65.0),(English,88
6 (Science,{(Science,90.0),(Science,89.0),(Science,78
7
8

```

```

1
2 grunt> illustrate records_group
3

```

```

-----
| student_marks | Subject:chararray | Student1:double | Student2:double | Student3:double | Student4:double |
|-----|-----|-----|-----|-----|-----|
|               | English           | 78.0             | 88.0             | 65.0             | 99.0             |
|-----|-----|-----|-----|-----|-----|
| bagged        | Subject:chararray | toPivot:bag{tuple(Student1:double)} |
|-----|-----|-----|-----|-----|
|               | English           | {(78.0), ..., (99.0)} |
|-----|-----|-----|-----|-----|
| pivoted_1     | Subject:chararray | toPivot::Student1:double |
|-----|-----|-----|-----|-----|
|               | English           | 78.0             |
|               | English           | 88.0             |
|               | English           | 65.0             |
|               | English           | 99.0             |
|-----|-----|-----|-----|-----|
| records_group | group:chararray   | pivoted_1:bag{tuple(Subject:chararray,toPivot::Student1:double)} |
|-----|-----|-----|-----|-----|
|               | English           | {(English, 78.0), ..., (English, 99.0)} |
|-----|-----|-----|-----|-----|

```

illustrate records_group

Step 8: Final result

```

1
2 grunt> result = FOREACH records_group GENERATE group
3

```



```
1  
2 grunt> dump result  
3  
  
1  
2 (Maths,92.0)  
3 (English,99.0)  
4 (Science,90.0)  
5  
  
1  
2 grunt> illustrate result  
3
```

[< 2. HBase Shell commands](#)[2. Apache Pig: Regex \(Regular expressions\) >](#)

Disclaimer

The contents in this Java-Success are copyrighted and from EmpoweringTech pty Ltd. The EmpoweringTech pty Ltd has the right to correct or enhance the current content without any prior notice. These are general advice only, and one needs to take his/her own circumstances into consideration. The EmpoweringTech pty Ltd will not be held liable for any damages caused or alleged to be caused either directly or indirectly by these materials and resources. Any trademarked names or labels used in this blog remain the property of their respective trademark owners. Links to external sites do not imply endorsement of the linked-to sites. [Privacy Policy](#).

© 2022 [java-success.com](https://www.java-success.com)

Top

