

**UNIVERSIDAD NACIONAL DE INGENIERÍA**

**FACULTAD DE CIENCIAS**



**CURSO** : SISTEMAS OPERATIVOS AVANZADOS  
**TEMA** : INSTALACIÓN DE PIG SOBRE CLUSTER  
MULTINODO  
**DOCENTE** : CESAR MARTIN CRUZ SALAZAR  
**ALUMNOS** : TOMAS J. CASAS RODRIGUEZ  
LUIS J. CARHUARICA AGUILAR

**LIMA – PERÚ**

**2016**

# INDICE

<b>1. INTRODUCCIÓN</b>	<b>3</b>
<b>1.1. PIG UNA HERRAMIENTA DE PROGRAMACIÓN</b>	
<b>1.2 PIG EN LA ACTUALIDAD</b>	
<b>2. ENTORNO</b>	<b>4</b>
<b>3. INSTALACIÓN</b>	<b>4</b>
<b>3.1 INSTALACION DE PIG</b>	
<b>3.2 INSTALACION DE APACHE ANT</b>	
<b>3.3 PRUEBAS</b>	
<b>4. APLICACION PARA PIG EN MODO SINGLE NODE</b>	<b>6</b>
<b>4.1 PIG LATIN</b>	
<b>EJEMPLO 1 : CARGA DE DATOS</b>	
<b>EJEMPLO 2 : CALCULAR MEDIA DE DATOS CON PIG</b>	
<b>EJEMPLO 3: ORDENANDO DATOS CON PIG</b>	
<b>4.2 . EJECUCION DE APLICACION MEDIANTE SCRIPT</b>	
<b>EJEMPLO 4 : USO EN UN SCRIPT QUE ALMACENA LA SALIDA EN UN FICHERO</b>	
<b>5. GRUNT SHELL PARA PIG EN MODO MULTINODO</b>	<b>14</b>
<b>EJEMPLO 5: MODO MULTINODO</b>	
<b>EJECUCION CON DOS NODOS</b>	
<b>RESULTADO</b>	
<b>LINK DE GITHUB</b>	
<b>6. CONCLUSIONES.</b>	<b>19</b>
<b>7. REFERENCIAS BIBLIOGRAFICAS</b>	<b>19</b>

# **INSTALACIÓN DE PIG SOBRE CLUSTER MULTINODO**

## **1. INTRODUCCIÓN.**

Apache Pig es una plataforma creada por Yahoo! que nos abstrae y simplifica el desarrollo de algoritmos MapReduce mediante una sintaxis parecida a SQL llamada Pig Latin. Tiene dos modos de funcionamiento por si queremos ejecutar sobre el cluster HDFS de Hadoop o en la máquina local.

Mediante un script se codifican las sentencias que realizan la carga, escaneo, búsqueda y filtrado de los datos de entrada y sentencias para el formateado y almacenamiento de los datos de salida. Estos datos pueden ser estructurados mediante un schema y así su acceso será más sencillo.[1]

### **1.1. PIG UNA HERRAMIENTA DE PROGRAMACIÓN**

Pig es una plataforma de alto nivel para crear programas MapReduce utilizados en Hadoop. El lenguaje de esta plataforma es llamado Pig Latin. Pig Latin abstrae la programación desde el lenguaje Java MapReduce en una notación que hace de MapReduce programación de alto nivel, similar a la de SQL para sistemas RDBMS.

Pig Latin puede ser ampliado utilizando UDF (Funciones Definidas por el Usuario) que el usuario puede escribir en Java, Python, Javascript, Ruby o Groovy y luego llamar directamente desde el lenguaje . Pig fue desarrollado originalmente por Yahoo Research en torno a 2006 por los investigadores para tener una forma ad-hoc de crear y ejecutar un trabajo map-reduce en conjuntos de datos muy grandes.[8]

En 2007, fue trasladado a Apache Software Foundation.

La capa de infraestructura de Pig se compone de un compilador que produce secuencias MapReduce, lo que permite a que los usuarios de Hadoop se enfoquen más en analizar los datos y dedicar menos tiempo en desarrollar aplicaciones MapReduce. El lenguaje de programación que utiliza Pig, Pig Latin, crea estructuras tipo SQL (SQL-like), de manera que, en lugar de escribir aplicaciones separadas de MapReduce, se pueda crear un script de Pig Latin el cual es automáticamente paralelizado y distribuido a través de un clúster.

### **1.2 PIG EN LA ACTUALIDAD**

Naturalmente Yahoo!, al ser creador de Pig, fue el primer usuario de la plataforma, tanto para los procesos de búsqueda web como al incorporarlo en Hadoop. De hecho, más de la mitad de procesos que son ejecutados en Hadoop están basados en scripts de Pig Latin. Pero no sólo Yahoo ha utilizado Pig; a partir del año 2009 otras compañías comenzaron a adoptar Pig dentro de su procesamiento de datos, algunas de ellas son:

“Gente que podrías conocer”, este componente de LinkedIn utiliza Hadoop y Pig para ofrecer recomendaciones de conocidos, páginas y empleos de interés.

“select count(\*) from tweets”, definitivamente SQL no es la opción para analizar tweets, retweets, usuarios, seguidores, etc., que conforman más de 12TB de información diaria. Twitter utiliza Pig para procesar estos logs de datos. [4]

AOL y WhitePages utilizan Pig para filtrar registros en sus procesos de búsqueda de información.

## 2. ENTORNO.

La instalacion se ha realizado con el siguiente entorno:

Raspbian usando la imagen 2016-05-27-raspbian-jessie.img .  
Oracle Java SDK version 7 .  
Apache Hadoop 2.7.2  
Apache Pig 0.15.0  
Apache Ant 1.9.7

## 3. INSTALACIÓN

### 3.1 INSTALACION DE PIG

Partimos de que en los raspberry ya tenemos instalado Apache Hadoop.

-Descargamos Pig desde la página de apache :

<http://apache.rediris.es/pig/pig-0.15.0/pig-0.15.0.tar.gz>

-Movemos a otra ruta

```
mv pig-0.15.0.tar.gz /usr/local  
cd /usr/local
```

-Descomprimos el .tar.gz

```
tar -xvf pig-0.15.0.tar.gz  
cd
```

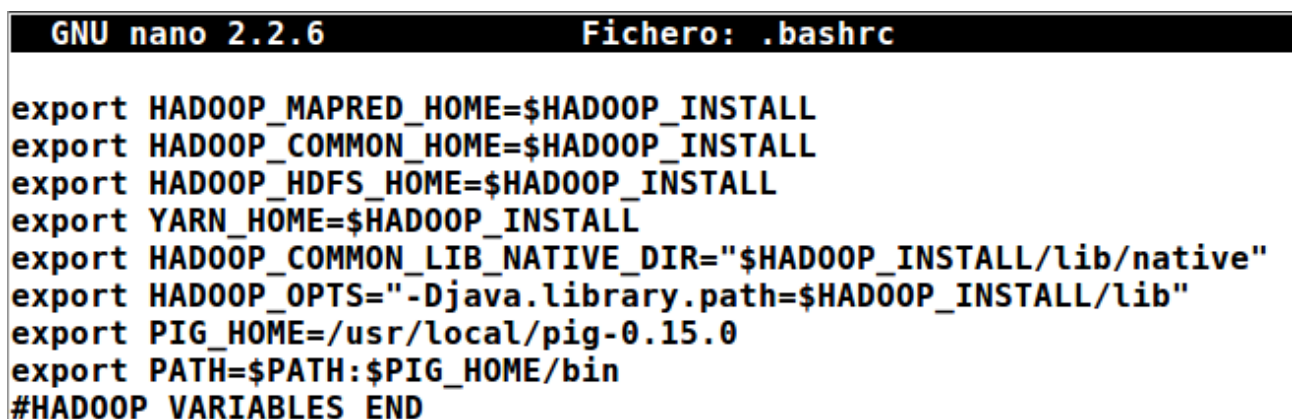
-Añadimos la variable de entorno PIG\_HOME en el \$HOME/.bashrc apuntando al directorio de instalación de Pig.

```
nano .bashrc
```

-Agregamos las siguientes lineas :

```
export PIG_HOME=/usr/local/pig-0.15.0  
export PATH=$PATH:$PIG_HOME/bin
```

-ctrl+o y ctrl+x para guardar y salir .



```
GNU nano 2.2.6           Fichero: .bashrc  
  
export HADOOP_MAPRED_HOME=$HADOOP_INSTALL  
export HADOOP_COMMON_HOME=$HADOOP_INSTALL  
export HADOOP_HDFS_HOME=$HADOOP_INSTALL  
export YARN_HOME=$HADOOP_INSTALL  
export HADOOP_COMMON_LIB_NATIVE_DIR="$HADOOP_INSTALL/lib/native"  
export HADOOP_OPTS="-Djava.library.path=$HADOOP_INSTALL/lib"  
export PIG_HOME=/usr/local/pig-0.15.0  
export PATH=$PATH:$PIG_HOME/bin  
#HADOOP VARIABLES END
```

-Cargamos de nuevo la configuración con :

```
source .bashrc
```

-Comprobamos que Pig está correctamente instalado indicando la versión :  
pig -version

```
hduser@NodoMaster:~$ source .bashrc
hduser@NodoMaster:~$ pig -version
Apache Pig version 0.15.0 (r1682971)
compiled Jun 01 2015, 11:44:35
hduser@NodoMaster:~$
```

### 3.2 INSTALACION DE APACHE ANT

-Por defecto Pig viene compilado para funcionar con versiones anteriores a Hadoop 2.X. Si partes con una versión 2.X de Hadoop debes compilar Pig indicando la versión de Hadoop, como es mi caso [5] . Para ello es necesario tener instalado Apache Ant, puedes descargarlo aquí. :

<http://www-us.apache.org/dist/ant/binaries/apache-ant-1.9.7-bin.tar.gz>

-Movemos el archivo  
mv apache-ant-1.9.7-bin.tar.gz /usr/local  
cd /usr/local

-Descomprimos  
tar -xvf apache-ant-1.9.7-bin.tar.gz

Las variables anteriores pueden ser definidas de dos maneras :

**Nivel Global:** Permite que las variables estén accesibles a todo usuario del sistema, en efecto permitiendo que cualquier usuario utilice apache-ant-1.9.7; estas definiciones son colocadas en el archivo /etc/profile del sistema.

**Nivel Usuario:** Las variables serían definidas únicamente para el usuario que utilice Ant; estas definiciones son colocadas en el archivo ~/.bashrc , donde ~/ es el directorio base del usuario.

-Hacemos  
sudo nano /etc/profile

```
GNU nano 2.2.6          Fichero: profile

    fi
done
unset i
fi

ANT_HOME="/usr/local/apache-ant-1.9.7"
PATH="$PATH:/usr/local/apache-ant-1.9.7/bin"
export ANT_HOME
export PATH
```

### 3.3 PRUEBAS

Para verificar la correcta instalación de Ant realice la siguiente prueba:

Invoque el comando `ant` de un directorio arbitrario (aleatorio) del sistema, si observa : Buildfile: build.xml does not exist! ha configurado

correctamente Ant, en caso contrario realice los pasos anteriores hasta que esta prueba sea ejecutada correctamente.

```
root@NodoMaster:/etc# ant
Buildfile: build.xml does not exist!
Build failed
root@NodoMaster:/etc#
```

Si todo es correcto, ejecutamos de nuevo `pig -version` .

**Para usar un ejemplo de Pig se debe tener Hadoop listo para su uso :**

-Preparando los directorios necesarios para Hadoop .

```
rm -rf /usr/local/hadoop_store/hdfs/datanode/current
```

```
rm -rf /usr/local/hadoop_store/hdfs/namenode/current
```

-Dando formato al Namenode

```
hdfs namenode -format
```

Se debe inicializar los demonios del hadoop

```
/usr/local/hadoop/sbin/start-dfs.sh && /usr/local/hadoop/sbin/start-yarn.sh && jps
```

## 4. APLICACION PARA PIG EN MODO SINGLE NODE <sup>[6]</sup>

**La aplicacion la descargamos de este link donde estan los ejemplos y los datos a usar :**

<https://github.com/jdcasasmoviles/pig-first-steps-master>

```
unzip pig-first-steps-master.zip
```

```
cd pig-first-steps-master
```

### 4.1 PIG LATIN

Para ir viendo la sintaxis de Pig crearemos algunos jobs MapReduce sobre un conjunto de datos.

Para ilustrar la simplicidad de Pig vamos a utilizar una fuente de datos que mapea los valores de las mediciones de una serie de estaciones climatológicas agrupadas por provincias de Castilla y León . En el reducer se calculaba la media del valor de la medida tomada pudiendo observar el dato de la contaminación de cada provincia recogido durante años de mediciones.[3]

#### **EJEMPLO 1 : CARGA DE DATOS**

Lo primero será hacer la carga de los datos ya que sin ellos no tendríamos nada que hacer. Creamos un fichero de texto llamado **ejemplo1.pig** y ponemos :

```

measure = load 'calidad_del_aire_cyl_1997_2013.csv' using PigStorage(';')
AS (date:chararray, co:float, no:float, no2:float, o3:float, pm10:float,
    sh2:float, pm25:float, pst:float, so2:float, province:chararray, station:chararray);

dump measure;
explain measure;
describe measure;

```

GNU nano 2.2.6

Fichero: ejemplo1.pig

- Hacemos la carga del fichero separando por ';' y creamos el schema.

```

measure = load 'calidad_del_aire_cyl_1997_2013.csv' using PigStorage(';') as (date:chararra
dump measure;
describe measure;
explain measure;

```

-Ejecucion de ejemplo1 .pig :Ejecutamos nuestro ejemplo con : pig -x local -f ejemplo1.pig

```

root@NodoMaster:/home/hduser/pig-first-steps-master# nano ejemplo1.pig
root@NodoMaster:/home/hduser/pig-first-steps-master# pig -x local -f ejemplo1.pig
16/10/31 21:37:08 INFO pig.ExecTypeProvider: Trying ExecType : LOCAL
16/10/31 21:37:08 INFO pig.ExecTypeProvider: Picked LOCAL as the ExecType
2016-10-31 21:37:08,261 [main] INFO org.apache.pig.Main - Apache Pig version 0.
15.0 (r1682971) compiled Jun 01 2015, 11:44:35

```

-Se mira como se cargan los datos :

```

(01/01/2011,,10.0,30.0,25.0,,,,,4.0,BURGOS,Medina de Pomar)
(02/01/2011,,5.0,18.0,36.0,,,,,4.0,BURGOS,Medina de Pomar)
(03/01/2011,,3.0,32.0,42.0,,,,,5.0,BURGOS,Medina de Pomar)
(04/01/2011,,8.0,30.0,27.0,,,,,4.0,BURGOS,Medina de Pomar)
(05/01/2011,,10.0,61.0,29.0,,,,,4.0,BURGOS,Medina de Pomar)
(06/01/2011,,8.0,79.0,30.0,,,,,4.0,BURGOS,Medina de Pomar)
(07/01/2011,,2.0,53.0,48.0,,,,,4.0,BURGOS,Medina de Pomar)
(08/01/2011,,2.0,49.0,51.0,,,,,2.0,BURGOS,Medina de Pomar)
(09/01/2011,,2.0,29.0,60.0,,,,,2.0,BURGOS,Medina de Pomar)
(10/01/2011,,4.0,40.0,36.0,,,,,2.0,BURGOS,Medina de Pomar)
(11/01/2011,,4.0,40.0,48.0,,,,,2.0,BURGOS,Medina de Pomar)
(12/01/2011,,9.0,41.0,37.0,,,,,2.0,BURGOS,Medina de Pomar)
(13/01/2011,,10.0,44.0,34.0,,,,,2.0,BURGOS,Medina de Pomar)
(14/01/2011,,9.0,43.0,30.0,,,,,2.0,BURGOS,Medina de Pomar)
(15/01/2011,,9.0,38.0,29.0,,,,,2.0,BURGOS,Medina de Pomar)
(16/01/2011,,6.0,43.0,35.0,,,,,2.0,BURGOS,Medina de Pomar)
(17/01/2011,,6.0,47.0,38.0,,,,,2.0,BURGOS,Medina de Pomar)
(18/01/2011,,4.0,25.0,39.0,,,,,2.0,BURGOS,Medina de Pomar)
(19/01/2011,,1.0,25.0,46.0,,,,,2.0,BURGOS,Medina de Pomar)
(20/01/2011,,2.0,17.0,51.0,,,,,2.0,BURGOS,Medina de Pomar)
(21/01/2011,,1.0,12.0,66.0,,,,,2.0,BURGOS,Medina de Pomar)
(22/01/2011,,1.0,6.0,68.0,,,,,1.0,BURGOS,Medina de Pomar)
(23/01/2011,,1.0,10.0,63.0,,,,,1.0,BURGOS,Medina de Pomar)
(24/01/2011,,1.0,13.0,57.0,,,,,1.0,BURGOS,Medina de Pomar)

```

```
|
| |---Project[bytearray][9] - scope-68
| |Cast[chararray] - scope-72
| |---Project[bytearray][10] - scope-71
| |Cast[chararray] - scope-75
| |---Project[bytearray][11] - scope-74
|
|--measure: Load(file:///home/hduser/pig-first-steps-master/calidad_del_aire_cyl_1997_2013.csv
PigStorage(';')) - scope-40-----
Global sort: false
-----

2016-10-31 21:40:00,881 [main] INFO  org.apache.pig.Main - Pig script completed in 29 seconds and 60
0 milliseconds (29660 ms)
root@NodoMaster:/home/hduser/pig-first-steps-master#
```

**load:** Indica que realice la carga de los datos del fichero `calidad_del_aire_cyl_1997_2013.csv`.  
**using PigStorage(“;”):** Indica que los datos deben ser separados por el carácter delimitador “;”. Esto lo hará la función `PigStorage` utilizada cuando tenemos un conjunto de datos estructurados y delimitados por algún carácter separador.[1]

<http://pig.apache.org/docs/r0.12.1/func.html#Load-Store-Functions>

**as:** Mediante ‘as’ definimos el schema de los datos cargados del fichero para acceder posteriormente a ellos de forma más sencilla. Indicamos a continuación el nombre de cada dato recogido y su tipo. Los tipos que admite son los tipos simples de Java: int, long, float, double, chararray, bytearray, boolean, datetime, biginteger, bigdecimal. También admite los tipos complejos: tuple (conjunto de campos ordenados), bag (una colección de tuplas), y map (conjunto de datos organizados por clave/valor).

**explain:** El operador 'explain' muestra el plan de ejecución de las tareas map reduce.

describe: El operador 'describe' saca por consola una descripción de la tupla generada.

En este caso describe el tipo de datos creado llamado ‘measure’. [3]

```
DIA;CO (mg/m3);NO (ug/m3);NO2 (ug/m3);O3 (ug/m3);PM10 (ug/m3);SH2 (ug/m3);PM25 (ug/m3);PS (ug/m3);SO2 (ug/m3);PROVINCIA;ESTACION
...
(16/06/2013,0.2,9.0,1.0,85.0,14.0,,,1.0,ZAMORA,Zamora 2)
(17/06/2013,0.2,11.0,1.0,67.0,8.0,,,2.0,ZAMORA,Zamora 2)
(18/06/2013,0.2,6.0,3.0,62.0,7.0,,,1.0,ZAMORA,Zamora 2)
```

8



```
filter_measure = filter measure by date != 'DIA';
```

Para realizar el cálculo que queremos obtener debemos agrupar por provincia, para eso Pig tiene la función 'group'.

```
measure_by_province = group filter_measure by province;
```

## **EJEMPLO 2 : CALCULAR MEDIA DE DATOS CON PIG**

Lo que tenemos ahora se puede parecer bastante a lo que tendríamos en la entrada del reducer, los valores del fichero agrupados por provincia. A continuación tendríamos que iterar sobre ellos y calcular la media. Pig tiene para eso la función 'avg'.

```
num_measures_by_province = foreach measure_by_province generate group, AVG(filter_measure.co) as measure;
```

Creamos el archivo ejemplo2.pig y ponemos los comandos expuestos.

GNU nano 2.2.6	Fichero: ejemplo2.pig	Modifi
<pre>-- Hacemos la carga del fichero separando por ';' y creamos el schema. measure = load 'calidad_del_aire_cyl_1997_2013.csv' using PigStorage(';') as (date:chararray,  --dump measure; --describe measure; --explain measure;  -- Filtramos los resultados, la primera línea no nos vale. filter_measure = filter measure by date != 'DIA';  -- Agrupamos los datos por provincia. measure_by_province = group filter_measure by province;  -- Recorremos los registros por provincia y calculamos la media de co. num_measures_by_province = foreach measure_by_province generate group, AVG(filter_measure.co)  dump num_measures_by_province;</pre>		

-Ejecucion de ejemplo2 .pig : Ejecutamos nuestro ejemplo con : pig -x local -f ejemplo2.pig

```
root@NodoMaster:/home/hduser/pig-first-steps-master# pig -x local -f ejemplo2.pig
16/10/31 22:01:03 INFO pig.ExecTypeProvider: Trying ExecType : LOCAL
16/10/31 22:01:03 INFO pig.ExecTypeProvider: Picked LOCAL as the ExecType
2016-10-31 22:01:03,933 [main] INFO org.apache.pig.Main - Apache Pig version 0.15.0 (r1682971
iled Jun 01 2015, 11:44:35
2016-10-31 22:01:03,934 [main] INFO org.apache.pig.Main - Logging error messages to: /home/hd
```

Si hacemos dump de num\_measures\_by\_province vemos el cálculo de la media por provincia:

```
2016-10-31 21:59:19,599 [main] INFO org.apache.pig.backend.hadoop.executionengine.util.MapRedU
Total input paths to process : 1
(LEÓN,0.9821986658627717)
(SORIA,0.18476821561128098)
(BURGOS,0.8641249233499702)
(ZAMORA,0.843978750701614)
(ÁVILA,0.9624001966559971)
(SEGOVIA,1.0198234750388033)
(PALENCIA,1.177672231415453)
(SALAMANCA,1.388050755824775)
(VALLADOLID,0.6850182736526162)
2016-10-31 21:59:19,655 [main] INFO org.apache.pig.Main - Pig script completed in 10 seconds a
6 milliseconds (10736 ms)
root@NodoMaster:/home/hduser/pig-first-steps-master#
```

### EJEMPLO 3 : ORDENANDO DATOS CON PIG

Si esto mismo lo estuviéramos haciendo con Hadoop, para ordenar la salida tendríamos que implementarnos nuestro propio tipo writable implementándonos el algoritmo de ordenación por el campo measure. En Pig bastaría con usar la función order by:

```
ordered_measures = order num_measures_by_province by measure;
```

Creamos el archivo ejemplo3.pig y ponemos los comandos expuestos.

GNU nano 2.2.6	Fichero: ejemplo3.pig	Modifi
<pre>-- Hacemos la carga del fichero separando por ';' y creamos el schema. measure = load 'calidad_del_aire_cyl_1997_2013.csv' using PigStorage(';') as (date:chararray,  --dump measure; --describe measure; --explain measure;  -- Filtramos los resultados, la primera línea no nos vale. filter_measure = filter measure by date != 'DIA';  -- Agrupamos los datos por provincia. measure_by_province = group filter_measure by province;  -- Recorremos los registros por provincia y calculamos la media de co. num_measures_by_province = foreach measure_by_province generate group, AVG(filter_measure.co)  dump num_measures_by_province;  -- Ordenamos de menor a mayor índice de co. ordered_measures = order num_measures_by_province by measure;  -- Mostramos los resultados ordenados. dump ordered_measures;</pre>		

-Ejecucion de ejemplo3 .pig : Ejecutamos nuestro ejemplo con : pig -x local -f ejemplo3.pig

```
Archivo Editar Ver Terminal Pestañas Ayuda
root@NodoMaster:/home/hduser/pig-first-steps-master# nano ejemplo3.pig
root@NodoMaster:/home/hduser/pig-first-steps-master# pig -x local -f ejemplo3.pig
16/10/31 22:18:44 INFO pig.ExecTypeProvider: Trying ExecType : LOCAL
16/10/31 22:18:44 INFO pig.ExecTypeProvider: Picked LOCAL as the ExecType
2016-10-31 22:18:44,679 [main] INFO org.apache.pig.Main - Apache Pig version 0.15.0 (r1682971
iled Jun 01 2015, 11:44:35
2016-10-31 22:18:44,680 [main] INFO org.apache.pig.Main - Logging error messages to: /home/hd
ig-first-steps-master/pig_1477970324678.log
2016-10-31 22:18:44,698 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - user.n
deprecated. Instead, use mapreduce.job.user.name
```

Si hacemos dump de ordered\_measures vemos el cálculo de la media por provincia ordenado de menor a mayor:

```
2016-10-31 22:19:00,529 [main] INFO org.apache.pig.backend.hadoop.executionengine.util.MapRed
Total input paths to process : 1
(SORIA,0.18476821561128098)
(VALLADOLID,0.6850182736526162)
(ZAMORA,0.843978750701614)
(BURGOS,0.8641249233499702)
(ÁVILA,0.9624001966559971)
(LEÓN,0.9821986658627717)
(SEGOVIA,1.0198234750388033)
(PALENCIA,1.177672231415453)
(SALAMANCA,1.388050755824775)
2016-10-31 22:19:00,570 [main] INFO org.apache.pig.Main - Pig script completed in 16 seconds
7 milliseconds (16577 ms)
root@NodoMaster:/home/hduser/pig-first-steps-master#
```

## 4.2 . EJECUCION DE APLICACION MEDIANTE SCRIPT

El código en Hadoop que hace este cálculo se compondría de un Mapper, un Reducer, un Custom Writable y un Driver que ejecute el Job. Todo eso se puede hacer de forma sencilla con el script de Pig que hemos creado en este ejemplo :

### EJEMPLO 4 : USO EN UN SCRIPT QUE ALMACENA LA SALIDA EN UN FICHERO

Almacenamos la salida en un fichero llamdo measures\_by\_province.out , mediante el siguiente comando :

```
store ordered_measures INTO 'measures_by_province.out';
```

Creamos el archivo ejemplo4.pig y ponemos los comandos expuestos.

```

- Hacemos la carga del fichero separando por ';' y creamos el schema.
measure = load 'calidad_del_aire_cyl_1997_2013.csv' using PigStorage(';') as (date:chararray, co:fl

--dump measure;
--describe measure;
--explain measure;

-- Filtramos los resultados, la primera línea no nos vale.
filter_measure = filter measure by date != 'DIA';

-- Agrupamos los datos por provincia.
measure_by_province = group filter_measure by province;

-- Recorremos los registros por provincia y calculamos la media de co.
num_measures_by_province = foreach measure_by_province generate group, AVG(filter_measure.co) as me

dump num_measures_by_province;

-- Ordenamos de menor a mayor índice de co.
ordered_measures = order num_measures_by_province by measure;

-- Mostramos los resultados ordenados.
dump ordered_measures;

-- Almacenamos la salida en un fichero
store ordered_measures INTO 'measures_by_province.out';

```

-Ejecucion de ejemplo4 .pig : Ejecutamos nuestro ejemplo con : pig -x local -f ejemplo4.pig

```

root@NodoMaster:/home/hduser/pig-first-steps-master# nano ejemplo4.pig
root@NodoMaster:/home/hduser/pig-first-steps-master# pig -x local -f ejemplo4.pig
16/10/31 22:30:11 INFO pig.ExecTypeProvider: Trying ExecType : LOCAL
16/10/31 22:30:11 INFO pig.ExecTypeProvider: Picked LOCAL as the ExecType
2016-10-31 22:30:11,417 [main] INFO org.apache.pig.Main - Apache Pig version 0.15.0 (r168297:
iled Jun 01 2015, 11:44:35
2016-10-31 22:30:11,418 [main] INFO org.apache.pig.Main - Logging error messages to: /home/hd
ig-first-steps-master/pig_1477971011415.log
2016-10-31 22:30:11,459 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - user.i
deprecated. Instead, use mapreduce.job.user.name

```

Veamos los resultados de la ejecucion del script en el fichero llamdo measures\_by\_province.out

```

root@NodoMaster:/home/hduser/pig-first-steps-master# ls
calidad_del_aire_cyl_1997_2013.csv  ejemplo4.pig          pig_1477968647828.log
ejemplo1.pig                       measures_by_province.out pig_1477971011415.log
ejemplo2.pig                       pig_1477967411330.log  README.md
ejemplo3.pig                       pig_1477967482477.log
root@NodoMaster:/home/hduser/pig-first-steps-master# cd measures_by_province.out
root@NodoMaster:/home/hduser/pig-first-steps-master/measures_by_province.out# ls
part-r-000000 SUCCESS
root@NodoMaster:/home/hduser/pig-first-steps-master/measures_by_province.out# nano part-r-000000

```

**Resultado :** Como se aprecia es el mismo resultado que el ejemplo3.pig solo que aquí esta en un fichero para una mejor apreciacion o uso , y se obtuvo con un script .

```
GNU nano 2.2.6                                Fichero: part-r-00000

SORIA      0.18476821561128098
VALLADOLID 0.6850182736526162
ZAMORA     0.843978750701614
BURGOS     0.8641249233499702
ÁVILA      0.9624001966559971
LEÓN       0.9821986658627717
SEGOVIA    1.0198234750388033
PALENCIA   1.177672231415453
SALAMANCA  1.388050755824775
█
```

-En nuestra interfaz de control del hadoop tenemos un nodo activo :

The screenshot shows the Hadoop cluster management interface. The top navigation bar includes 'Nodes of the cluster' and a search bar. The main content area is titled 'Nodes of the cluster' and displays 'Cluster Metrics' and 'Scheduler Metrics'. The 'Cluster Metrics' table shows various resource usage statistics. The 'Scheduler Metrics' table shows the 'Capacity Scheduler' with a 'MEMORY' resource type and a minimum allocation of '<memory:1024, vCores:1>'. Below these, a table lists the nodes in the cluster, including 'Node Labels', 'Rack', 'Node State', 'Node Address', 'Node HTTP Address', 'Last health-update', 'Health-report', and 'Containers'. The 'Node State' column shows 'RUNNING' for the 'NodoMaster:37641' node, which is also labeled as 'NodoMaster:8042'.

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reserved	Active Nodes	De
0	0	0	0	0	0 B	8 GB	0 B	0	8	0	1	0

Scheduler Type	Scheduling Resource Type	Minimum Allocation
Capacity Scheduler	[MEMORY]	<memory:1024, vCores:1>

Node Labels	Rack	Node State	Node Address	Node HTTP Address	Last health-update	Health-report	Containers
/default-rack		RUNNING	NodoMaster:37641	NodoMaster:8042	Mon Oct 31 22:43:07 -0500 2016		0

Showing 1 to 1 of 1 entries

The screenshot shows the Hadoop cluster management interface, specifically the 'NodeManager' details page. The top navigation bar includes 'Nodes of the cluster' and a search bar. The main content area is titled 'NodeManager' and displays 'NodeManager information'. The 'NodeManager information' table shows various resource usage statistics. The 'NodeManager information' table shows the 'Total Vmem allocated for Containers' as 16.80 GB, 'Vmem enforcement enabled' as true, 'Total Pmem allocated for Container' as 8 GB, 'Pmem enforcement enabled' as true, 'Total Vcores allocated for Containers' as 8, 'NodeHealthyStatus' as true, 'LastNodeHealthTime' as 'Mon Oct 31 22:43:07 PET 2016', 'NodeHealthReport' as 'Node Manager Version: 2.7.2 from b165c4fe8a74265c792ce23f546c64604acf0e41 by jenkins sol checksum c63f7cc71b8f63249e35126f0f7492d on 2016-01-26T00:16Z', and 'Hadoop Version: 2.7.2 from b165c4fe8a74265c792ce23f546c64604acf0e41 by jenkins sol checksum d0fda26633fa762bff87ec759ebe689c on 2016-01-26T00:08Z'.

Total Vmem allocated for Containers	Vmem enforcement enabled	Total Pmem allocated for Container	Pmem enforcement enabled	Total Vcores allocated for Containers	NodeHealthyStatus	LastNodeHealthTime	NodeHealthReport
16.80 GB	true	8 GB	true	8	true	Mon Oct 31 22:43:07 PET 2016	Node Manager Version: 2.7.2 from b165c4fe8a74265c792ce23f546c64604acf0e41 by jenkins sol checksum c63f7cc71b8f63249e35126f0f7492d on 2016-01-26T00:16Z
Hadoop Version: 2.7.2 from b165c4fe8a74265c792ce23f546c64604acf0e41 by jenkins sol checksum d0fda26633fa762bff87ec759ebe689c on 2016-01-26T00:08Z							



## 5. GRUNT SHELL PARA PIG EN MODO MULTINODO

Pig tiene dos modos de funcionamiento: en local que es el single node y en el HDFS que es el multinodo . Para ejecutar pig en local lo hacemos mediante la instrucción `pig -x local` como vimos anteriormente . Esto iniciará el shell de Pig llamado Grunt.

Desde este shell podemos ir escribiendo las instrucciones anteriores sin necesidad de crear el script. Cada instrucción será recogida y compilada y desencadenará la creación de un Job para ejecutar la tarea sobre Hadoop.[2]

En el grunt shell también disponemos de instrucciones para trabajar con el sistema de ficheros HDFS, el shell de unix, etc.

**MODO MULTINODO :** En el modo HDFS el script de Pig será distribuido en el cluster de Hadoop. Para ello debemos invocar al grunt shell mediante la instrucción `pig` .

```
hduser@NodoMaster:~$ pig
16/11/01 03:22:27 INFO pig.ExecTypeProvider: Trying ExecType : LOCAL
16/11/01 03:22:27 INFO pig.ExecTypeProvider: Trying ExecType : MAPREDUCE
16/11/01 03:22:27 INFO pig.ExecTypeProvider: Picked MAPREDUCE as the ExecType
2016-11-01 03:22:27,163 [main] INFO org.apache.pig.Main - Apache Pig version 0.15.0 (r1682971) compiled Jun 01 2015, 11:44:35
2016-11-01 03:22:27,163 [main] INFO org.apache.pig.Main - Logging error messages to: /home/hduser/pig_1477988547161.log
2016-11-01 03:22:27,259 [main] INFO org.apache.pig.impl.util.Utils - Default bootstrap file /home/hduser/.pigbootstrap not found
2016-11-01 03:22:28,222 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.address
2016-11-01 03:22:28,222 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
2016-11-01 03:22:28,223 [main] INFO org.apache.pig.backend.hadoop.executionengine.HExecutionEngine - Connecting to hadoop file system at: hdfs://NodoMaster:54310
2016-11-01 03:22:28,668 [main] WARN org.apache.hadoop.util.NativeCodeLoader - Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
2016-11-01 03:22:30,737 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.address
2016-11-01 03:22:30,738 [main] INFO org.apache.pig.backend.hadoop.executionengine.HExecutionEngine - Connecting to map-reduce job tracker at: NodoMaster:54311
2016-11-01 03:22:30,741 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
grunt>
grunt>
```

### EJEMPLO 5: MODO MULTINODO

Salimos del grunt con el comando `:quit` . Para que funcione nuestro ejemplo debemos subir al HDFS el fichero de entrada de datos. Primero creamos el directorio a usar esto lo hacemos en la terminal normal del sistema :

```
hdfs dfs -mkdir /user
hdfs dfs -mkdir /user/hduser
hdfs dfs -chmod -R 777 /user/hduser
hdfs dfs -mkdir /tmp
hdfs dfs -chmod -R 777 /tmp
```

Luego en el grunt shell :

```
2016-11-01 03:50:59,955 [main] INFO org.apache.ame is deprecated. Instead, use fs.defaultFS
grunt> mkdir ejemplo5
grunt> █
```

Esto habrá creado el directorio en el HDFS. Para comprobarlo ejecutamos:

```
grunt> ls
hdfs://NodoMaster:54310/user/root/ejemplo5
grunt> █
```

Nos ubicamos en el directorio pig-first-steps-master .

```
hduser@NodoMaster:~$ ls
Descargas  hadoop-2.7.2  Música  pig-first-steps-master  Público
Documentos Imágenes      pig_1477988547161.log pig-first-steps-master.zip smallfile.txt
Escritorio mediumfile.txt pig_1477989050699.log Plantillas  Vídeos
hduser@NodoMaster:~$ cd pig-first-steps-master
hduser@NodoMaster:~/pig-first-steps-master$ ls
calidad_del_aire_cyl_1997_2013.csv  ejemplo4.pig  pig_1477968647828.log
ejemplo1.pig                       measures_by_province.out  pig_1477971011415.log
ejemplo2.pig                       pig_1477967411330.log    README.md
ejemplo3.pig                       pig_1477967482477.log
hduser@NodoMaster:~/pig-first-steps-master$ █
```

Hacemos el comando pig y subimos al HDFS el fichero de datos:

```
grunt> copyFromLocal calidad_del_aire_cyl_1997_2013.csv ejemplo5
grunt> █
```

El único cambio en el script que tenemos que hacer es en la ruta al load del fichero y en la salida, hay que cambiarla por ejemplo5/calidad\_del\_aire\_cyl\_1997\_2013.csv y por ejemplo5/measures\_by\_province.out .

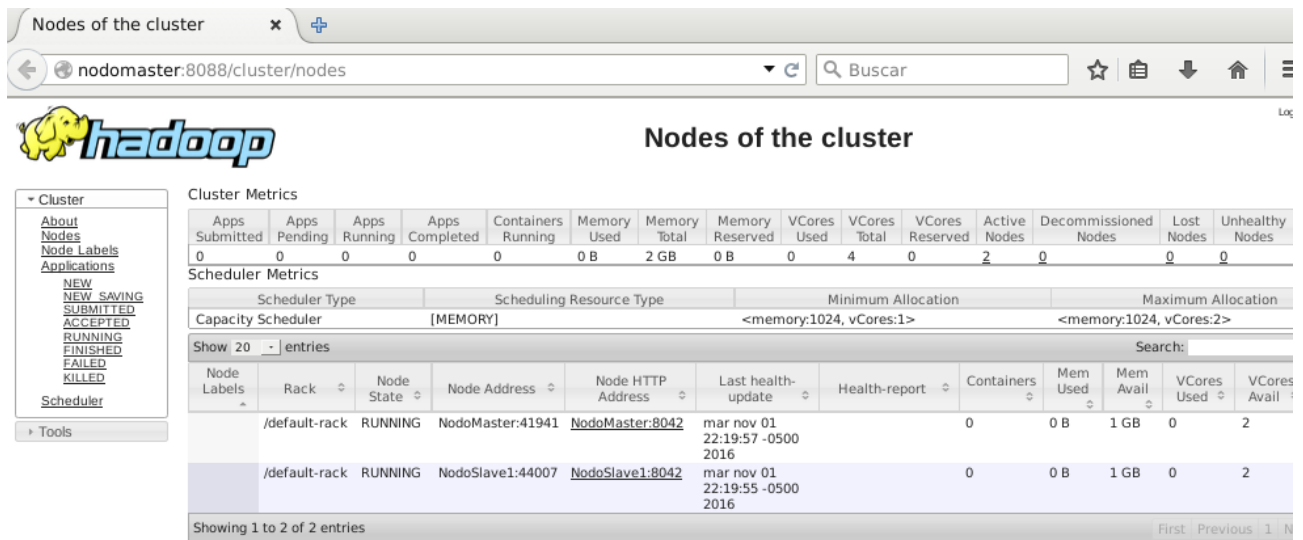
GNU nano 2.2.6	Fichero: ejemplo5.pig	Modificado
<pre>-- Hacemos la carga del fichero separando por ';' y creamos el schema. measure = load 'ejemplo5/calidad_del_aire_cyl_1997_2013.csv' using PigStorage(';') as (date:chararr\$  --dump measure; --describe measure; --explain measure;  -- Filtramos los resultados, la primera línea no nos vale. filter_measure = filter measure by date != 'DIA';  -- Agrupamos los datos por provincia. measure_by_province = group filter_measure by province;  -- Recorremos los registros por provincia y calculamos la media de co. num_measures_by_province = foreach measure_by_province generate group, AVG(filter_measure.co) as me\$  dump num_measures_by_province;  -- Ordenamos de menor a mayor índice de co. ordered_measures = order num_measures_by_province by measure;  -- Mostramos los resultados ordenados. dump ordered_measures;  -- Almacenamos la salida en un fichero store ordered_measures INTO 'ejemplo5/measures_by_province.out';█</pre>		

Para ejecutar el ejemplo en el HDFS podemos copiar línea a línea el script y ejecutándolo en el grunt shell o bien desde fuera del grunt shell mediante la instrucción:

```
hduser@NodoMaster:~/pig-first-steps-master$ pig -f ejemplo5.pig
16/11/01 04:13:11 INFO pig.ExecTypeProvider: Trying ExecType : LOCAL
16/11/01 04:13:11 INFO pig.ExecTypeProvider: Trying ExecType : MAPREDUCE
16/11/01 04:13:11 INFO pig.ExecTypeProvider: Picked MAPREDUCE as the ExecType
16/11/01 04:13:11 WARN pig.Main: Cannot write to log file: /home/hduser/pig-first-step
477991591417.log
```

## EJECUCION CON DOS NODOS :

En el navegador ponemos : <http://192.168.1.2:8088/cluster/nodes>



The screenshot shows the Hadoop cluster management web interface. The browser address bar displays 'nodomaster:8088/cluster/nodes'. The page title is 'Nodes of the cluster'. On the left, there is a sidebar with a 'Cluster' menu containing links like 'About', 'Nodes', 'Node Labels', 'Applications', and 'Scheduler'. The main content area displays 'Cluster Metrics' and 'Scheduler Metrics'. Below these, there is a table showing the status of nodes in the cluster.

Cluster Metrics														
Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reserved	Active Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes
0	0	0	0	0	0 B	2 GB	0 B	0	4	0	2	0	0	0

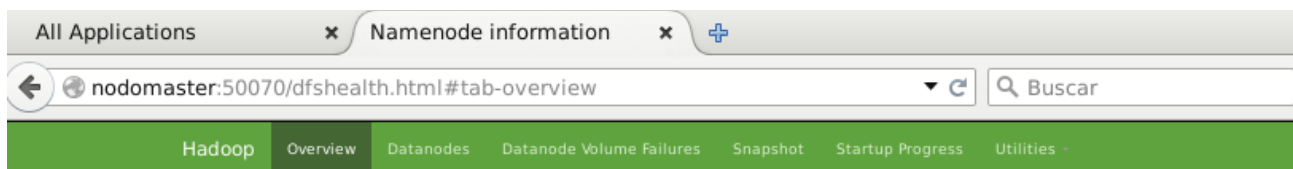
  

Scheduler Metrics													
Scheduler Type		Scheduling Resource Type		Minimum Allocation		Maximum Allocation							
Capacity Scheduler		[MEMORY]		<memory:1024, vCores:1>		<memory:1024, vCores:2>							
Showing 1 to 2 of 2 entries													
Node Labels	Rack	Node State	Node Address	Node HTTP Address	Last health-update	Health-report	Containers	Mem Used	Mem Avail	VCores Used	VCores Avail		
/default-rack		RUNNING	NodoMaster:41941	NodoMaster:8042	mar nov 01 22:19:57 -0500 2016		0	0 B	1 GB	0	2		
/default-rack		RUNNING	NodoSlave1:44007	NodoSlave1:8042	mar nov 01 22:19:55 -0500 2016		0	0 B	1 GB	0	2		

Showing 1 to 2 of 2 entries

Podemos comprobar accediendo al NameNode si esta activo :

<http://192.168.1.2:50070/dfshealth.html>



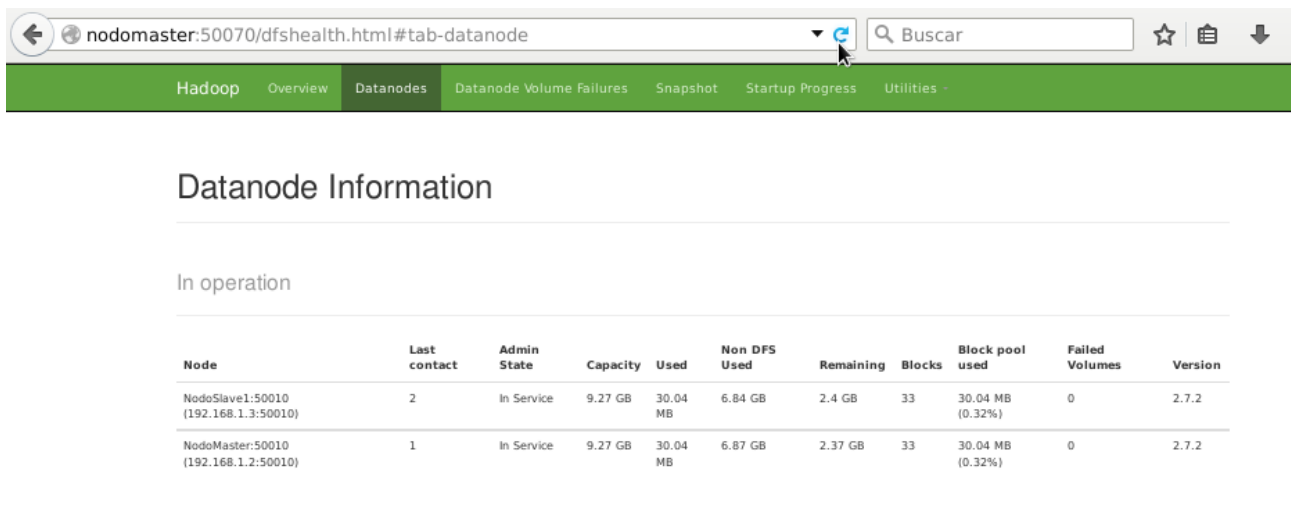
The screenshot shows the Hadoop NameNode information web interface. The browser address bar displays 'nodomaster:50070/dfshealth.html#tab-overview'. The page title is 'NameNode information'. The interface has a green header bar with tabs: 'Hadoop', 'Overview', 'Datanodes', 'Datanode Volume Failures', 'Snapshot', 'Startup Progress', and 'Utilities'. The 'Overview' tab is selected.

## Overview 'NodoMaster:54310' (active)

Started:	Tue Nov 01 22:34:37 PET 2016
Version:	2.7.2, rb165c4fe8a74265c792ce23f546c64604acf0e41
Compiled:	2016-01-26T00:08Z by jenkins from (detached from b165c4f)
Cluster ID:	CID-8911baab-0cd5-41e8-a201-10ccc0830216
Block Pool ID:	BP-1747739726-192.168.1.2-1478057620322



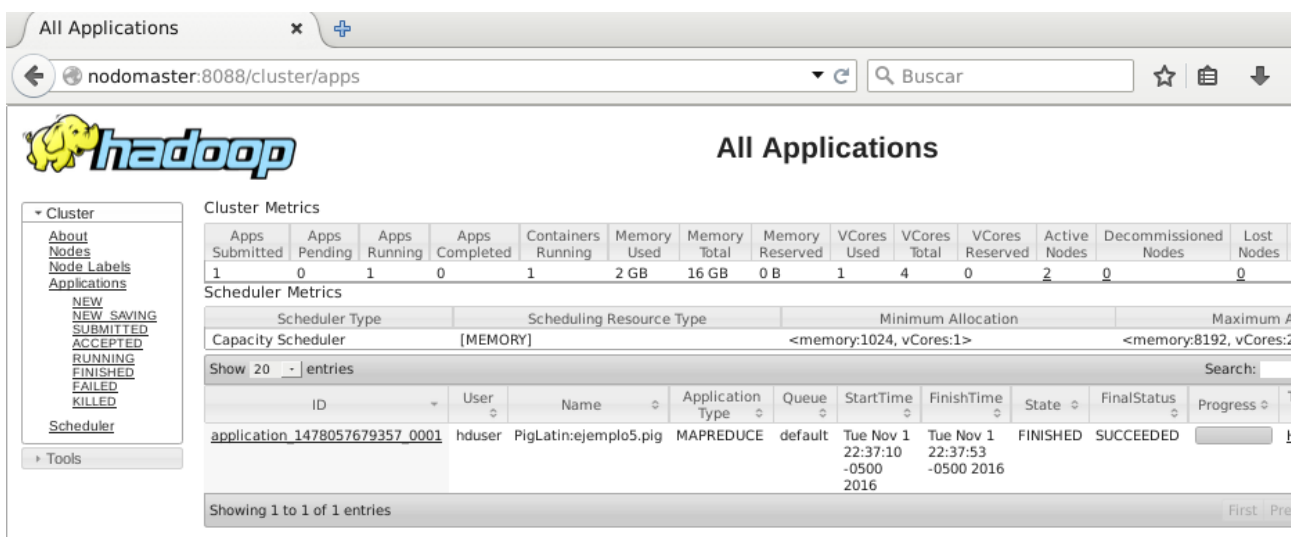
DataNodes :



The screenshot shows the 'Datanodes' tab in the Hadoop DFS Health interface. The browser address bar shows 'nodomaster:50070/dfshealth.html#tab-datanode'. The page title is 'Datanode Information'. Below the title, it says 'In operation'. A table lists the datanodes with columns: Node, Last contact, Admin State, Capacity, Used, Non DFS Used, Remaining, Blocks, Block pool used, Failed Volumes, and Version.

Node	Last contact	Admin State	Capacity	Used	Non DFS Used	Remaining	Blocks	Block pool used	Failed Volumes	Version
NodoSlave1:50010 (192.168.1.3:50010)	2	In Service	9.27 GB	30.04 MB	6.84 GB	2.4 GB	33	30.04 MB (0.32%)	0	2.7.2
NodoMaster:50010 (192.168.1.2:50010)	1	In Service	9.27 GB	30.04 MB	6.87 GB	2.37 GB	33	30.04 MB (0.32%)	0	2.7.2

Se van ejecutando las tareas que nuestro script tiene :



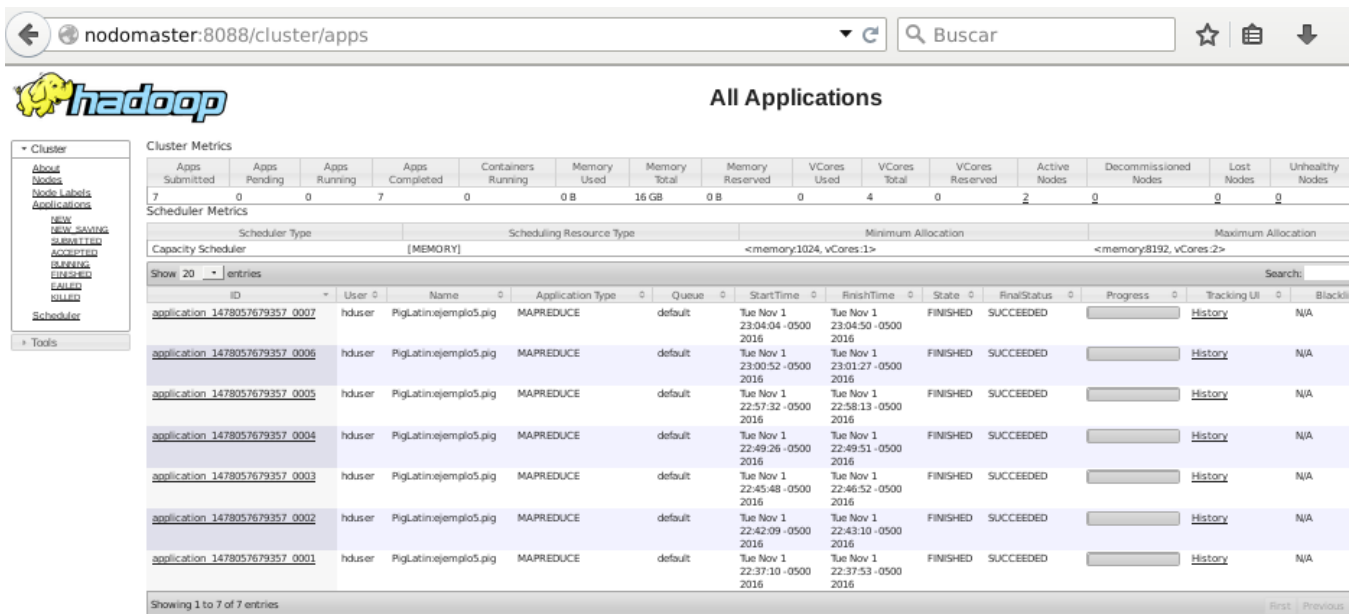
The screenshot shows the 'All Applications' page in the Hadoop interface. The browser address bar shows 'nodomaster:8088/cluster/apps'. The page title is 'All Applications'. On the left, there is a sidebar with navigation links. The main content area shows 'Cluster Metrics' and 'Scheduler Metrics'. Below these, there is a table of applications with columns: ID, User, Name, Application Type, Queue, StartTime, FinishTime, State, FinalStatus, and Progress.

ID	User	Name	Application Type	Queue	StartTime	FinishTime	State	FinalStatus	Progress
application_1478057679357_0001	hduser	PigLatin:ejemplo5.pig	MAPREDUCE	default	Tue Nov 1 22:37:10 -0500 2016	Tue Nov 1 22:37:53 -0500 2016	FINISHED	SUCCEEDED	100%

Procesamiento de los datos , viendo informacion mediante terminal se aprecia que va un 83 % de completado .

```
?016-11-01 23:04:05,051 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.Ma
?ReduceLauncher - HadoopJobId: job_1478057679357_0007
?016-11-01 23:04:05,051 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.Ma
?ReduceLauncher - Processing aliases ordered_measures
?016-11-01 23:04:05,051 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.Ma
?ReduceLauncher - detailed locations: M: ordered_measures[20,19] C: R:
?016-11-01 23:04:24,335 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.Ma
?ReduceLauncher - 83% complete
?016-11-01 23:04:24,335 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.Ma
```

-Tareas finalizadas , apreciacion mediante la interfaz de Hadoop .



The screenshot shows the Hadoop cluster management interface. At the top, there's a navigation bar with the URL 'nodomaster:8088/cluster/apps' and a search bar. Below this, the 'All Applications' page is displayed. On the left, there's a sidebar with navigation links like 'Cluster', 'About Nodes', 'Node Labels', 'Applications', 'NEW', 'NEW SAVING', 'SUBMITTING', 'ACCEPTED', 'RUNNING', 'FINISHED', 'KILLED', 'SCHEDULED', and 'Tools'. The main content area shows 'Cluster Metrics' and 'Scheduler Metrics'. The 'Cluster Metrics' table includes columns for Apps Submitted, Pending, Running, Completed, Containers Running, Memory Used, Memory Total, Memory Reserved, V-Cores Used, V-Cores Total, V-Cores Reserved, Active Nodes, Decommissioned Nodes, Lost Nodes, and Unhealthy Nodes. The 'Scheduler Metrics' table includes columns for Scheduler Type, Scheduling Resource Type, Minimum Allocation, and Maximum Allocation. Below these, there's a table of application entries with columns for ID, User, Name, Application Type, Queue, Start Time, Finish Time, State, Final Status, Progress, Tracking UI, and Backfill. The table shows several application entries, all with a state of 'FINISHED' and a final status of 'SUCCEEDED'.

-El procesamiento se completo se puede apreciar **Pig script completed en la terminal .**

```
2016-11-01 23:12:29,753 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLa
pReduceLauncher - Success!
2016-11-01 23:12:29,928 [main] INFO org.apache.pig.Main - Pig script completed
conds and 963 milliseconds (2125963 ms)
hduser@NodoMaster:~/pig-first-steps-master$ hdfs dfs -ls /user/hduser/ejemplo5/measures_by_pro
out
16/11/01 23:13:53 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your pl
... using builtin-java classes where applicable
Found 2 items
-rw-r--r-- 2 hduser supergroup 0 2016-11-01 23:04 /user/hduser/ejemplo5/measures_by
nce.out/_SUCCESS
-rw-r--r-- 2 hduser supergroup 240 2016-11-01 23:04 /user/hduser/ejemplo5/measures_by
nce.out/part-r-00000
```

**RESULTADO :** Para ver el resultado , ejecutamos los comandos :

```
hdfs dfs -ls /user/hduser/ejemplo5/measures_by_province.out
hdfs dfs -cat /user/hduser/ejemplo5/measures_by_province.out/part-r-00000
```

```
hduser@NodoMaster:~/pig-first-steps-master$ hdfs dfs -cat /user/hduser/ejemplo5/measures_by_p
.out/part-r-00000
16/11/01 23:14:31 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your p
... using builtin-java classes where applicable
SORIA 0.18476821561128098
VALLADOLID 0.6850182736526162
ZAMORA 0.843978750701614
BURGOS 0.8641249233499702
ÁVILA 0.9624001966559971
LEÓN 0.9821986658627717
SEGOVIA 1.0198234750388033
PALENCIA 1.177672231415453
SALAMANCA 1.388050755824775
hduser@NodoMaster:~/pig-first-steps-master$
```

Se aprecia que el resultado es el mismo que obtuvimos antes , solo que esta vez lo obtuvimos con Pig en modo multinodo y con Hadoop .

**Link de github con los paquetes , ejemplos y datos usados en este trabajo. :**

<https://github.com/jdcasasmoviles/pig-first-steps-master>

## **6. CONCLUSIONES.**

1-Se procesa enormes datos con pig de una manera muy rapida y la ventaja es que ya tiene configurado distintos modos de uso ya sea local o multinodo para el uso en un cluster , con esto se procesa datos en tiempos cortos .

2-Este lenguaje tiene muchas posibilidades ya que podemos definirnos nuestras propias funciones que realizan tareas más complejas .

3- Apache Pig, un lenguaje que bien usado puede ahorrarnos muchísimo tiempo a la hora de procesar nuestros trabajos MapReduce.

## **7. REFERENCIAS BIBLIOGRAFICAS**

[1]<https://www.adictosaltrabajo.com/tutoriales/pig-first-steps/>

[2]<https://pig.apache.org/docs/r0.7.0/setup.html>

[3]<http://www.ibm.com/developerworks/ssa/library/l-apachepigdataquery/>

[4]<http://docplayer.es/12856589-Comprender-un-poco-mas-de-los-que-es-apache-pig-y-hadoop-el-tutorial-de-cerdo-muestra-como-ejecutar-dos-scripts-de-cerdo-en-modo-local-y-el.html>

[5][https://xml.osmosislatina.com/ant\\_linux.htm](https://xml.osmosislatina.com/ant_linux.htm)

[6]<https://www.youtube.com/watch?v=GdqrGMiH6Xc>

[7]<http://pig.apache.org/docs/r0.12.1/start.html>

[8][https://es.wikipedia.org/wiki/Pig\\_\(herramienta\\_de\\_programaci%C3%B3n\)](https://es.wikipedia.org/wiki/Pig_(herramienta_de_programaci%C3%B3n))

## **ANEXOS**

### **CONFIGURACION DE RED**

#### **CONFIGURACION DE RED SLAVE**

```
nano /etc/network/interfaces
    auto lo
    iface lo inet loopback

    #adaptador eth0
    auto eth0
    iface eth0 inet static
    address 192.168.1.3
    netmask 255.255.255.0
    gateway 192.168.1.252

sudo nano /etc/hostname
    NodoSlave1

nano /etc/hosts
    127.0.0.1    localhost
    127.0.1.1    NodoSlave1
    # The following lines are desirable for IPv6 capable hosts
    ::1    localhost ip6-localhost ip6-loopback
    ff02::1 ip6-allnodes
    ff02::2 ip6-allrouters
    192.168.1.2 NodoMaster
    192.168.1.3 NodoSlave1

nano /etc/dhcpd.conf
    interface eth0
    static ip_address=192.168.1.3/24
    static routers=192.168.1.252
    static domain_name_servers=200.48.225.130 200.48.225.146

nano /etc/resolv.conf
    search Home
    nameserver 200.48.225.130
    nameserver 200.48.225.146
```

#### **CONFIGURACION DE RED MASTER**

```
nano /etc/network/interfaces
    auto lo
    iface lo inet loopback

    #adaptador eth0
    auto eth0
    iface eth0 inet static
    address 192.168.1.2
    netmask 255.255.255.0
    gateway 192.168.1.252

sudo nano /etc/hostname
    NodoMaster

nano /etc/hosts
    127.0.0.1    localhost
    127.0.1.1    NodoMaster
    # The following lines are desirable for IPv6 capable hosts
    ::1    localhost ip6-localhost ip6-loopback
    ff02::1 ip6-allnodes
    ff02::2 ip6-allrouters
    192.168.1.2 NodoMaster
    192.168.1.3 NodoSlave1
```

```
nano /etc/dhcpd.conf
    interface eth0
        static ip_address=192.168.1.2/24
        static routers=192.168.1.252
        static domain_name_servers=200.48.225.130 200.48.225.146
```

```
nano /etc/resolv.conf
    search Home
    nameserver 200.48.225.130
    nameserver 200.48.225.146
```

### **CONFIGURACION DE ARCHIVOS DE HADOOP : SINGLE NODO**

```
sudo chown -R hduser:hadoop /usr/local/hadoop
su hduser
cd
nano .bashrc
    #HADOOP VARIABLES START
    export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-armhf
    #export JAVA_HOME=/usr/lib/jvm/java-1.7.0-openjdk-armhf
    export HADOOP_INSTALL=/usr/local/hadoop
    export PATH=$PATH:$HADOOP_INSTALL/bin
    export PATH=$PATH:$HADOOP_INSTALL/sbin
    export HADOOP_MAPRED_HOME=$HADOOP_INSTALL
    export HADOOP_COMMON_HOME=$HADOOP_INSTALL
    export HADOOP_HDFS_HOME=$HADOOP_INSTALL
    export YARN_HOME=$HADOOP_INSTALL
    export HADOOP_COMMON_LIB_NATIVE_DIR="$HADOOP_INSTALL/lib/native"
    export HADOOP_OPTS="-Djava.library.path=$HADOOP_INSTALL/lib"
    #HADOOP VARIABLES END
```

#### **hadoop-env.sh**

```
nano /usr/local/hadoop/etc/hadoop/hadoop-env.sh
```

```
export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-armhf
# Extra Java runtime options. Empty by default.
export HADOOP_OPTS="$HADOOP_OPTS -Djava.net.preferIPv4Stack=true -server"
```

#### **mapred-env.sh**

```
nano /usr/local/hadoop/etc/hadoop/mapred-env.sh
export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-armhf
```

#### **HDFS**

```
sudo mkdir -p /usr/local/hadoop_store/hdfs/namenode
sudo mkdir -p /usr/local/hadoop_store/hdfs/datanode
sudo chown -R hduser:hadoop /usr/local/hadoop_store
nano /usr/local/hadoop/etc/hadoop/hdfs-site.xml
```

```
<configuration>
<property>
  <name>dfs.permissions</name>
  <value>>false</value>
</property>
<property>
  <name>dfs.replication</name>
  <value>1</value>
  <description>Default block replication.
  The actual number of replications can be specified when the file is created.
  The default is used if replication is not specified in create time.
  </description>
</property>
<property>
  <name>dfs.namenode.name.dir</name>
  <value>file:/usr/local/hadoop_store/hdfs/namenode</value>
```

```

</property>
<property>
  <name>dfs.datanode.data.dir</name>
  <value>file:/usr/local/hadoop_store/hdfs/datanode</value>
</property>
</configuration>

```

## CORE

```

sudo mkdir -p /app/hadoop/tmp
sudo chown hduser:hadoop /app/hadoop/tmp
nano /usr/local/hadoop/etc/hadoop/core-site.xml

```

```

<configuration>
<property>
  <name>hadoop.tmp.dir</name>
  <value>/app/hadoop/tmp</value>
  <description>A base for other temporary directories.</description>
</property>

```

```

<property>
  <name>fs.default.name</name>
  <value>hdfs://NodoMaster:54310</value>
  <description>The name of the default file system. A URI whose
scheme and authority determine the FileSystem implementation. The
uri's scheme determines the config property (fs.SCHEME.impl) naming
the FileSystem implementation class. The uri's authority is used to
determine the host, port, etc. for a filesystem.</description>
</property>
</configuration>

```

## MAPRED

```

cp /usr/local/hadoop/etc/hadoop/mapred-site.xml.template /usr/local/hadoop/etc/hadoop/mapred-site.xml
nano /usr/local/hadoop/etc/hadoop/mapred-site.xml

```

```

<configuration>
<property>
  <name>mapred.job.tracker</name>
  <value>NodoMaster:54311</value>
  <description>The host and port that the MapReduce job tracker runs
at. If "local", then jobs are run in-process as a single map
and reduce task.
</description>
</property>
</configuration>

```

## YARN

```

nano /usr/local/hadoop/etc/hadoop/yarn-site.xml

```

```

<configuration>
<property>
  <name>yarn.nodemanager.aux-services</name>
  <value>mapreduce_shuffle</value>
</property>
</configuration>

```

## CONFIGURACION DE ARCHIVOS DE HADOOP : MULTINODO EN EL NODO SLAVE

### HDFS

```

nano /usr/local/hadoop/etc/hadoop/hdfs-site.xml

```

```

<configuration>
<property>
  <name>dfs.permissions.enabled</name>
  <value>>false</value>
</property>

```

```

<property>
  <name>dfs.replication</name>

  <value>2</value>
  <description>Default block replication.
  The actual number of replications can be specified when the file is created.
  The default is used if replication is not specified in create time.
  </description>
</property>
<property>
  <name>dfs.namenode.name.dir</name>
  <value>file:/usr/local/hadoop_store/hdfs/namenode</value>
</property>
<property>
  <name>dfs.datanode.data.dir</name>
  <value>file:/usr/local/hadoop_store/hdfs/datanode</value>
</property>
</configuration>

```

## CORE

nano /usr/local/hadoop/etc/hadoop/core-site.xml

```

<configuration>
<property>
  <name>hadoop.tmp.dir</name>
  <value>/app/hadoop/tmp</value>
  <description>A base for other temporary directories.</description>
</property>
<property>
  <name>fs.defaultFS</name>
  <value>hdfs://NodoMaster:54310</value>
  <description>The name of the default file system. A URI whose
  scheme and authority determine the FileSystem implementation. The
  uri's scheme determines the config property (fs.SCHEME.impl) naming
  the FileSystem implementation class. The uri's authority is used to
  determine the host, port, etc. for a filesystem.</description>
</property>
</configuration>

```

## YARN

nano /usr/local/hadoop/etc/hadoop/yarn-site.xml

```

<configuration>
<property>
  <name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
  <value>org.apache.hadoop.mapreduce.ShuffleHandler</value>
</property>
<property>
  <name>yarn.nodemanager.aux-services</name>
  <value>mapreduce_shuffle</value>
</property>
<property>
  <name>yarn.resourcemanager.scheduler.address</name>
  <value>NodoMaster:8030</value>
</property>
<property>
  <name>yarn.resourcemanager.address</name>
  <value>NodoMaster:8032</value>
</property>
<property>
  <name>yarn.resourcemanager.webapp.address</name>
  <value>NodoMaster:8088</value>
</property>
</configuration>

```

```

<name>yarn.resourcemanager.resource-tracker.address</name>
<value>NodoMaster:8031</value>
</property>
<property>
<name>yarn.nodemanager.resource.cpu-vcores</name>
<value>2</value>
</property>
</configuration>

```

## MAPRED

nano /usr/local/hadoop/etc/hadoop/mapred-site.xml

```

<configuration>
  <property>
    <name>mapred.job.tracker</name>
    <value>NodoMaster:54311</value>
  </property>
  <property>
    <name>mapreduce.framework.name</name>
    <value>yarn</value>
  </property>
</configuration>

```

## NODO MASTER

### HDFS

nano /usr/local/hadoop/etc/hadoop/hdfs-site.xml

```

<configuration>
<property>
  <name>dfs.permissions.enabled</name>
  <value>>false</value>
</property>
<property>
  <name>dfs.replication</name>
  <value>2</value>
  <description>Default block replication.
The actual number of replications can be specified when the file is created.
The default is used if replication is not specified in create time.
</description>
</property>
<property>
  <name>dfs.namenode.name.dir</name>
  <value>file:/usr/local/hadoop_store/hdfs/namenode</value>
</property>
<property>
  <name>dfs.datanode.data.dir</name>
  <value>file:/usr/local/hadoop_store/hdfs/datanode</value>
</property>
</configuration>

```

### CORE

nano /usr/local/hadoop/etc/hadoop/core-site.xml

```

<configuration>
<property>
  <name>hadoop.tmp.dir</name>
  <value>/app/hadoop/tmp</value>
  <description>A base for other temporary directories.</description>
</property>

<property>
  <name>fs.defaultFS</name>
  <value>hdfs://NodoMaster:54310</value>
  <description>The name of the default file system. A URI whose
scheme and authority determine the FileSystem implementation. The

```



```

uri's scheme determines the config property (fs.SCHEME.impl) naming
the FileSystem implementation class. The uri's authority is used to
determine the host, port, etc. for a filesystem.</description>
</property>
</configuration>

```

## YARN

```
nano /usr/local/hadoop/etc/hadoop/yarn-site.xml
```

```

<configuration>
<property>
<name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
<value>org.apache.hadoop.mapreduce.ShuffleHandler</value>
</property>
<property>
<name>yarn.nodemanager.aux-services</name>
<value>mapreduce_shuffle</value>
</property>
<property>
<name>yarn.resourcemanager.scheduler.address</name>
<value>NodoMaster:8030</value>
</property>
<property>
<name>yarn.resourcemanager.address</name>
<value>NodoMaster:8032</value>
</property>
<property>
<name>yarn.resourcemanager.webapp.address</name>
<value>NodoMaster:8088</value>
</property>
<property>
<name>yarn.resourcemanager.resource-tracker.address</name>
<value>NodoMaster:8031</value>
</property>
<property>
<name>yarn.nodemanager.resource.cpu-vcores</name>
<value>2</value>
</property>
</configuration>

```

## MAPRED

```
nano /usr/local/hadoop/etc/hadoop/mapred-site.xml
```

```

<configuration>
<property>
<name>mapred.job.tracker</name>
<value>NodoMaster:54311</value>
<description>The host and port that the MapReduce job tracker runs
at. If "local", then jobs are run in-process as a single map
and reduce task.
</description>
</property>
<property>
<name>mapreduce.framework.name</name>
<value>yarn</value>
</property>
</configuration>

```

## ANTES DE EJECUTAR HACER EN NODO MASTER

```

rm -rf /usr/local/hadoop_store/hdfs/datanode/current
rm -rf /usr/local/hadoop_store/hdfs/namenode/current
hdfs namenode -format
/usr/local/hadoop/sbin/start-dfs.sh
/usr/local/hadoop/sbin/start-yarn.sh && jps

```

