

¿Qué ofrece Autentia Real Business Solutions S.L?

Somos su empresa de **Soporte a Desarrollo Informático**.
 Ese apoyo que siempre quiso tener...

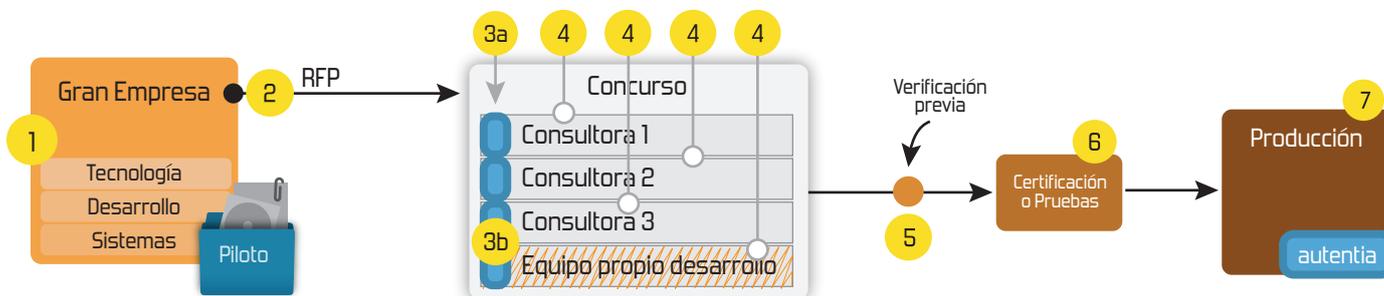
1. Desarrollo de componentes y proyectos a medida



2. Auditoría de código y recomendaciones de mejora

3. Arranque de proyectos basados en nuevas tecnologías

1. Definición de frameworks corporativos.
2. Transferencia de conocimiento de nuevas arquitecturas.
3. Soporte al arranque de proyectos.
4. Auditoría preventiva periódica de calidad.
5. Revisión previa a la certificación de proyectos.
6. Extensión de capacidad de equipos de calidad.
7. Identificación de problemas en producción.



4. Cursos de formación (impartidos por desarrolladores en activo)

Spring MVC, JSF-PrimeFaces /RichFaces,
 HTML5, CSS3, JavaScript-jQuery

Control de autenticación y
 acceso (Spring Security)
 UDDI

JPA-Hibernate, MyBatis
 Motor de búsqueda empresarial (Solr)
 ETL (Talend)

Gestor portales (Liferay)
 Gestor de contenidos (Alfresco)
 Aplicaciones híbridas

Web Services
 Rest Services
 Social SSO
 SSO (Cas)

Dirección de Proyectos Informáticos.
 Metodologías ágiles
 Patrones de diseño
 TDD

Tareas programadas (Quartz)
 Gestor documental (Alfresco)
 Inversión de control (Spring)

BPM (jBPM o Bonita)
 Generación de informes (JasperReport)
 ESB (Open ESB)

AdictosAlTrabajo

Temporada Completa de Terrakos
terrakos.com



Entra en Adictos a través de

E-mail:

Contraseña:

[Deseo registrarme](#)
[Olvíde mi contraseña](#)

- [Inicio](#)
- [Quiénes somos](#)
- [Formación](#)
- [Comparador de salarios](#)
- [Nuestros libros](#)
- [Más](#)

» Estás en: [Inicio](#) [Tutoriales](#) Primeros pasos con Apache Pig



Juan Alonso Ramos

Consultor tecnológico de desarrollo de proyectos informáticos.

Ingeniero en Informática, especialidad en Ingeniería del Software

Puedes encontrarme en Autentia: Ofrecemos de servicios soporte a desarrollo, factoría y formación

Somos expertos en Java/J2EE

[Ver todos los tutoriales del autor](#)

Version: Latest
OS: Mac OSX
Price: Free

Download

This advertisement will lead you to our website where you can download Genio

Fecha de publicación del tutorial: 2014-04-23

Tutorial visitado 4 veces [Descargar en PDF](#)

Primeros pasos con Apache Pig

0. Índice de contenidos.

- 1. Introducción.
- 2. Entorno.
- 3. Instalación.
- 4. Pig Latin.
- 5. Grunt Shell.
- 6. Conclusiones.

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.

Si crees que el desarrollo de algoritmos MapReduce es una tarea compleja, te invito a que sigas leyendo este tutorial donde vamos a ver cómo podemos simplificar estas tareas aprendiendo el lenguaje que nos aporta Apache Pig.

Puedes descargarte el código del tutorial desde mi repositorio de github pinchando [aquí](#).

2. Entorno.

El tutorial se ha realizado con el siguiente entorno:

- Ubuntu 12.04 64 bits
- Oracle Java SDK 1.6.0_27
- Apache Hadoop 2.2.0
- Apache Pig 0.12.0
- Apache Ant 1.7.1

3. Instalación

Partimos de que en la máquina ya tenemos instalado Apache Hadoop. Si no fuera así podéis seguir el tutorial de [primeros pasos con Hadoop: instalación y configuración en Linux](#)

Descargamos Pig desde la [página de apache](#).

Descomprimos el .tar.gz

```
1 | tar -xvf pig-0.12.1.tar.gz
2 | mv pig-0.12.1.tar.gz /usr/local/pig-0.12.1
```

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

Catálogo de servicios Autentia



Síguenos a través de:



Últimas Noticias

» [Aprende gratis ReactiveCocoa](#)

» [Checklist de Scrum de Autentia](#)

» [Autentia estrena web y celebra el X Cycling Day](#)

» [Buscamos personal para Autentia y nuestros clientes \(10-Marzo-2014\)](#)

» [Charla de Auto Layout en nuestra oficina](#)

[Histórico de noticias](#)

Últimos Tutoriales

» [Android Flavors](#)

» [Maven, Jenkins, Sonar y tests de integración](#)

» [Spring Ldap: gestión de transacciones y configuración de un pool de conexiones](#)

» [Spring Ldap: operaciones básicas haciendo uso del soporte de plantillas](#)

» [Crear un plugin para Android](#)

██████████
[Page Pushers](#)
[Community](#)
[Help?](#)

██████████
0 people brought clicks to this page

no clicks
+
+
+
+
+
+
+
+

powered by [karmacracy](#)

en PhoneGap

```
1 | export PIG_HOME=/usr/local/pig-0.12.1
2 | export PATH=$PATH:$PIG_HOME/bin
```

Cargamos de nuevo la configuración con `source .bashrc` y comprobamos que Pig está correctamente instalado indicando la versión 0.12.1.

```
1 | pig -version
```

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. Para ello es necesario tener instalado Apache Ant, puedes descargarlo [aquí](#). La última versión cuando se escribió este tutorial es la 1.9.3. También lo puedes instalar mediante el instalador de paquetes de Ubuntu el cual os instalará la versión 1.7 que será suficiente para lo que vamos a necesitar.

Desde el directorio de instalación de pig, `/usr/local/pig-0.12.1` ejecutamos la tarea:

```
1 | ant clean jar-all -Dhadoopversion=23
```

Si todo es correcto, ejecutamos de nuevo `pig -version` y nos indicará que es la versión 0.12.1-SNAPSHOT.

4. 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 la misma fuente de datos que en el tutorial de [primeros pasos de MapReduce con Hadoop](#). En ese tutorial veíamos un job de Hadoop que mapeaba 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.

Lo primero será hacer la carga de los datos ya que sin ellos no tendríamos nada que hacer. Creamos un fichero de texto llamado `air-quality.pig`:

```
1 | measure = load 'calidad_del_aire_cyl_1997_2013.csv' using PigStorage(';')
2 | AS (date:chararray, co:float, no:float, no2:float, o3:float, pm10:float,
3 |    sh2:float, pm25:float, pst:float, so2:float, province:chararray, station:chararray);
4 |
5 | dump measure;
6 | explain measure;
7 | describe measure;
```

Con sólo estas pocas líneas, Pig es capaz de recorrer el fichero y cargar los datos. Vamos a repasar con más detalle cómo lo hace:

- **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. Existen otras funciones de carga y almacenamiento como `BinStorage`, `TextStorage`, `JsonLoader`, `HbaseStorage`. Más info [aquí](#).
- **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`, `bigint`, `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).
- **dump**: Usamos el operador de diagnóstico 'dump' para visualizar los datos recogidos en la carga anterior. Es útil para sacar los datos por pantalla.
- **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'.

Un trozo del dump resultante:

```
1 | DIA;CO (mg/m3);NO (ug/m3);NO2 (ug/m3);O3 (ug/m3);PM10 (ug/m3);SH2 (ug/m3);PM25 (ug/m3);?:
2 | ...
3 | (16/06/2013,0.2,9.0,1.0,85.0,14.0,,,1.0,ZAMORA,Zamora 2)
4 | (17/06/2013,0.2,11.0,1.0,67.0,8.0,,,2.0,ZAMORA,Zamora 2)
5 | (18/06/2013,0.2,6.0,3.0,62.0,7.0,,,1.0,ZAMORA,Zamora 2)
```

Una vez que hemos hecho la carga de los datos desde nuestro fichero y almacenado en tipos de datos más manejables a través de la definición de un schema, vamos a realizar el cálculo de la media de monóxido de carbono (co), primer valor del conjunto de datos. Aunque antes de eso, vamos a quitar la cabecera del fichero ya que ocupa la primera línea y es algo que no nos interesa. Para ello utilizamos la función `'filter'` indicando que nos quite la línea que contiene en la variable definida como 'date' el valor 'DIA'. El resultado nos lo quedamos en el alias `'filter_measure'`.

```
1 | 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'`

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

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'`.

```
1 | num_measures_by_province = foreach measure_by_province generate group, AVG(filter_mesu?:
```

Últimos Tutoriales del Autor

- » [Implementando tu propio Writable en Hadoop](#)
- » [Primeros pasos de MapReduce con Hadoop](#)
- » [Primeros pasos con Hadoop: instalación y configuración en Linux](#)
- » [Trabajando con Mule ESB](#)
- » [Crear un proyecto de Mule ESB con Mule Studio](#)

Si hacemos dump de `num_measures_by_province` vemos el cálculo de la media por provincia:

```
1 | (LEÓN,0.9821986658627717) ?
2 | (SORIA,0.18476821561128098)
3 | (BURGOS,0.8641249233499702)
4 | (ZAMORA,0.843978750701614)
5 | (ÁVILA,0.9624001966559971)
6 | (SEGOVIA,1.0198234750388033)
7 | (PALENCIA,1.177672231415453)
8 | (SALAMANCA,1.388050755824775)
9 | (VALLADOLID,0.6850182736526162)
```

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`:

```
1 | ordered_measures = order num_measures_by_province by measure; ?
```

Si hacemos dump de `ordered_measures` vemos el cálculo de la media por provincia ordenado de menor a mayor:

```
1 | (SORIA,0.18476821561128098) ?
2 | (VALLADOLID,0.6850182736526162)
3 | (ZAMORA,0.843978750701614)
4 | (BURGOS,0.8641249233499702)
5 | (ÁVILA,0.9624001966559971)
6 | (LEÓN,0.9821986658627717)
7 | (SEGOVIA,1.0198234750388033)
8 | (PALENCIA,1.177672231415453)
9 | (SALAMANCA,1.388050755824775)
```

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 en sólo 6 líneas.

Para ejecutar en local el script que hemos creado lo hacemos mediante la siguiente instrucción:

```
1 | pig -x local -f air-quality.pig ?
```

5. Grunt Shell

Pig tiene dos modos de funcionamiento: en local y en el HDFS. Para ejecutar pig en local lo hacemos mediante la instrucción `pig -x local`. 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.

En el grunt shell también disponemos de instrucciones para trabajar con el sistema de ficheros HDFS, el shell de unix, etc. Podemos ver la ayuda mediante `help`

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`

```
1 | pig ?
```

Para que funcione nuestro ejemplo debemos subir al HDFS el fichero de entrada de datos. Primero creamos el directorio

```
1 | grunt> mkdir air-quality ?
```

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

```
1 | grunt> ls ?
2 |
3 | hdfs://localhost:8020/user/hadoop/air-quality <dir>
4 | </dir>
```

Subimos al HDFS el fichero de datos:

```
1 | grunt> copyFromLocal calidad_del_aire_cyl_1997_2013.csv air-quality ?
```

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 `air-quality/calidad_del_aire_cyl_1997_2013.csv` y por `air-quality/measures_by_province.out`

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:

```
1 | pig -f air-quality.pig ?
```

Podemos comprobar accediendo al cliente del NameNode `http://localhost:50070/dfshealth.jsp` que el resultado es el mismo que vimos anteriormente.

6. Conclusiones.

En este tutorial hemos visto un poquito de lo que es Apache Pig, un lenguaje que bien usado puede ahorrarnos muchísimo tiempo a la hora de procesar nuestros trabajos MapReduce. Este lenguaje tiene muchas posibilidades ya que podemos definir nuestras propias funciones que realizan tareas más complejas, pero eso lo veremos en otro tutorial.

Toda la información está disponible en la [documentación oficial](#).

Puedes descargar el código del tutorial desde mi repositorio de github pinchando [aquí](#).

Espero que te haya sido de ayuda.

Un saludo.

Juan

A continuación puedes evaluarlo:

[Regístrate para evaluarlo](#)

Por favor, vota +1 o compártelo si te pareció interesante

Share |



Ánimate y coméntanos lo que pienses sobre este **TUTORIAL**:

» **Regístrate** y accede a esta y otras ventajas «



Esta obra está licenciada bajo licencia Creative Commons de Reconocimiento-No comercial-Sin obras derivadas 2.5

Copyright 2003-2014 © All Rights Reserved | [Texto legal y condiciones de uso](#) | [Banners](#) | [Powered by Autentia](#) | [Contacto](#)

