

**UNIVERSIDAD DE SANTIAGO DE CHILE**  
**FACULTAD DE INGENIERÍA**  
**DEPARTAMENTO DE INGENIERÍA INFORMÁTICA**

# **Hadoop**

**Integrantes:** **Diego Herrera**  
**Diego Olavarría**  
**Victor Reyes**  
**Profesor:** **Roberto González**  
**Curso:** **Sistemas Distribuidos**

Septiembre, 2015

## Introducción

Estamos viviendo la época de la revolución del Big Data, donde los grandes volúmenes de datos, usados para trabajar, han superado con creces, la capacidad de procesamiento de un simple host. El Big Data nace para solucionar estos problemas:

- Cómo almacenar y trabajar con grandes volúmenes de datos.
- Y la más importante, como poder interpretar y analizar estos datos, de naturaleza muy dispar.

Si miramos alrededor nuestro, vemos que cualquier dispositivo que usamos genera datos, estos pueden ser analizados actualmente. De esta gran cantidad de datos que tenemos a nuestro alcance, sólo el 20% se trata de información estructura y el 80% son datos no estructurados. Estos últimos añaden complejidad en la forma que se tienen que almacenar y analizar.

Hadoop aparece en el mercado como una solución para estos problemas, dando una forma de almacenar y procesar estos datos, a través del cloud computing.

## Cloud Computing

La computación en la nube, de acuerdo con un documento especial publicado bajo los auspicios del NIST , es “un modelo para permitir de manera conveniente el acceso ubicuo a la red bajo demanda, a un conjunto compartido de recursos informáticos configurables (por ejemplo, redes, servidores, almacenamiento, aplicaciones y servicios) que pueden ser rápidamente accedidos y liberados con un esfuerzo de gestión mínimos o proveedor de servicios de interacción ” .

Otra definición se pueden encontrar en un documento oficial salió de Comisión de la UE: Una "nube" es un entorno de ejecución elástica de los recursos que participan múltiples partes interesadas y la prestación de un servicio medido en múltiples granularidad para un determinado nivel de calidad ( de servicio) .

La IEEE la define como un algoritmo que almacena la información en servidores de internet de manera permanente y se envía a cachés temporales de los clientes.

El servicio de nube tiene la ventaja de hacer parecer al cliente, que tiene recursos infinitos. Esto requiere que el manejo y asignación de recursos de la nube se haga de manera automática. La virtualización es la solución a los problemas de escalabilidad que esto provoca.

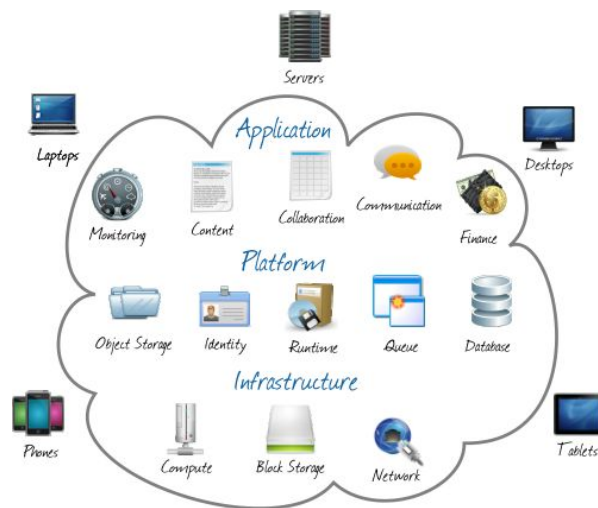
Al ejecutar múltiples máquinas virtuales en la nube, las CPU, memoria principal y dispositivos de red de un mismo servidor son compartidas, lo que lleva a cuellos de botella en los bus de entrada/salida.

Los modelos más relevantes de nubes son los siguientes:

1. Nube pública: Los recursos de estas están disponibles para el público en general
2. Nube de comunidad: Permite a los miembros de una comunidad compartir la infraestructura de una nube. Los miembros de esta nube tienen intereses, objetivos y metas comunes.
3. Nube híbrida: Posee características tanto de una nube pública y de una privada.
4. Nube privada: Este tipo de nube es manejado por una sola entidad, siendo esta misma quien maneja y mantiene la nube.

La infraestructura de la nube se encuentra en tres modelos de servicio:

- Software como servicio: Esta infraestructura permite que los clientes no necesiten instalar las aplicaciones en sus propios computadores, sino hace que éstas corran en la nube en su totalidad, a cambio de no tener que manejar más que simples configuraciones sobre éstas.
- Plataforma como servicio: Los proveedores con este servicio proveen un entorno total para aplicaciones webs, servidores webs, herramientas de desarrollo y base de datos en la nube, permitiendo a clientes que se encuentren separados poder trabajar sobre la mismo servicio.
- Infraestructura como servicio: ofrece servicios de cómputo y almacenamiento como servicio estándar en la web.



**Virtualización:** Es la creación de recursos a través de un software, como por ejemplo una plataforma de hardware, un sistema operativo, un dispositivo de almacenamiento u otros recursos de red.

## Hadoop

Hadoop es un framework que ejecuta aplicaciones en grandes clusters de hardware dedicado. Este framework proporciona a las aplicaciones fiabilidad y movilidad de datos de forma transparente. Hadoop implementa un paradigma computacional llamado *MapReduce*, donde la aplicación se divide en muchos pequeños fragmentos de trabajo, cada uno de los cuales se pueden ejecutar o volver a ejecutar en cualquier nodo del clúster. Además, proporciona un sistema de archivos distribuido que almacena los datos en los nodos de cómputo, produciendo un alto ancho de banda agregado en todo el clúster. Ambos, *MapReduce* y el sistema de archivos distribuidos, están diseñados de manera que las fallas de nodo se gestionan automáticamente mediante el framework.

El marco Hadoop está implementado en una arquitectura maestro-esclavo utilizado tanto para el almacenamiento de datos distribuidos como para cálculos distribuidos. Hadoop corre en un clúster completamente configurado que se compone de un conjunto de demonios o módulos internos ejecutados en varias máquinas del clúster. Estos demonios tienen tareas específicas: Algunos de ellos corren solamente en una sola máquina, mientras que otros que poseen las instancias se ejecutan en varias máquinas. Los demonios principales de Hadoop son:

## HDFS

El HDFS o sistema de distribución de archivos de Hadoop provee un acceso global a los archivos en un cluster, maximizando la portabilidad, para esto se explota los sistemas de archivos nativo de cada nodo.

- NameNode

El demonio más importante. Daemon NameNode es el controlador principal de HDFS (sistema de archivos distribuido para el framework hadoop), mantiene el directorio raíz y coordina los demonios DataNode esclavo para la realización de operaciones de entrada/salida, o sea, sabe el nombre de los archivos, donde están y de cuantos bloques se compone. Éste es el único punto de fallo crítico de la agrupación.

Los clientes pueden contactar el namenode para abrir, cerrar, renombrar y borrar archivos.

- DataNode

Cada máquina esclavo perteneciente al clúster será el anfitrión de un demonio DataNode, leyendo y escribiendo bloques HDFS de los archivos almacenados en el sistema de archivos local, no se necesita que los bloques estén en un mismo sistema de archivos locales porque este se abstrae de eso.

- NameNode Secundaria (Secondary NameNode)

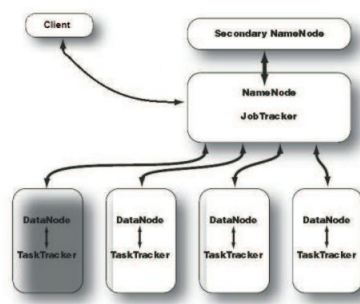
Tiene la función de supervisar el estado de salud de los clúster HDFS. Cada grupo tiene un NameNode y un NameNode secundario que se ejecutan en máquinas separadas, con la principal diferencia entre los dos demonios de que el último no recibe cambios en tiempo real dentro de los HDFS.

- JobTracker

Asegura la sincronización entre la aplicación y el marco Hadoop. JobTracker establece el plan de ejecución en concordancia con los archivos que se van a procesar, asigna nodos para cada tarea y supervisa todas las tareas en ejecución. Cada clúster Hadoop tiene un solo demonio JobTracker, que se ejecuta en un servidor como nodo maestro.

- TaskTracker

Es responsable de llevar a cabo las tareas asignadas por JobTracker. Cada nodo esclavo tiene un solo TaskTracker que pueden iniciar varias JVM para ejecutar más tareas en paralelo.



Una muestra de la arquitectura de clúster Hadoop se representa en la figura. Para un clúster pequeño, el demonio NameNode Secundario puede estar alojado en un nodo esclavo mientras que, en un gran grupo, se acostumbra a separar los demonios NameNode y JobTracker en diferentes máquinas. En la figura, tenemos un nodo maestro corriendo los demonios NameNode y JobTracker y un nodo independiente que aloja el demonio NameNode secundaria, sólo en caso de fallo del nodo maestro.

Cada máquina esclavo alberga dos demonios, un DataNode y TaskTracker, con el fin de ejecutar tareas de cálculo en el mismo nodo donde se almacenan sus datos de entrada.

## YARN

Mapreduce desde la versión 0.23 de Hadoop fue actualizado a MapReduce 2 (MRv2) o también llamado YARN.

Con esto el JobTracker fue separado en 2, el ResourceManager (RM) que se encarga de manejar los recursos y tiempos de vida de los nodos de manera global por cluster y el ApplicationMaster (AM) que maneja el plan lógico de un simple trabajo por petición de

recursos del RM. Aparte cada nodo pasa a tener un NodeManager que se comunica con el ResourceManager, siendo este último el corazón de la escalabilidad de esta arquitectura.

- ResourceManager (RM)

Este demonio expone 2 interfaces públicas, la suscripción de clientes a aplicaciones y la negociación con el AM por acceso a recursos, por otro lado tiene una interfaz interna dirigida al NM que monitorea los clusters y el manejo de recursos.

El AM codifica sus necesidades de recursos en uno o más ResourceRequest, cada cual tiene número de contenedores, recursos por contenedor, preferencias locales y prioridad de las solicitudes dentro de la aplicación.

- ApplicationManager (AM)

Una aplicación puede ser un set de procesos estáticos, una descripción lógica de trabajo o un servicio de larga duración. El AM es el proceso encargado de coordinar las ejecuciones de las aplicaciones en el cluster y a la vez corre como una en este. El AM periódicamente conversa con el RM para avisarle que está vivo y actualizar el registro de las demandas.

- NodeManager (NM)

Maneja la asignación y manejo de contenedores, monitoreando y proveyendo un set de servicios a estos, la memoria, CPU y otros recursos disponibles son registrados por el operador en el YARN, después de registrados en el RM, el NM queda en espera de instrucciones y envía sus datos cada cierto tiempo.

Hadoop se puede utilizar en los siguientes modos :

- Independiente: No hay demonios lanzados. Todos los procesos se ejecutan en una sola máquina virtual. Se utiliza para desarrollo, prueba y depuración.
- Pseudo - distribuido: Los demonios Hadoop se están ejecutando en el equipo local, pero en máquinas virtuales independientes, simulando un grupo real. Las comunicaciones entre máquinas representadas por el tráfico de red están ausentes.
- Distribuido: Los demonios Hadoop se ejecutan en un clúster.

## Implementación

Estos pasos deben realizarse en todas las máquinas que se utilizarán en el clúster. Cuando se especifique, algunos comandos se utilizarán solo en la máquina maestra, en la máquina esclavo, o en todas.

## Instalacion Java

Como primer paso se debe instalar Java de ser necesario, ya que Hadoop corre sobre éste. Para esto en Ubuntu se usan los siguientes comandos para la instalación en la consola:

```
master@master:~$ sudo apt-get install python-software-properties
master@master:~$ sudo add-apt-repository ppa:ferranroberto/java
```

El segundo comando comprende el uso de otro repositorio recomendado para el uso de Hadoop en Ubuntu.

Se procede a actualizar Ubuntu:

```
master@master:~$ sudo apt-get update
```

E instalar los paquetes de Java:

```
master@master:~$ sudo apt-get install default-jre
master@master:~$ sudo apt-get install default-jdk
master@master:~$ sudo apt-get install oracle-java8-installer
master@master:~$ sudo apt-get install sun-java6-jdk
```

```
$ sudo update-java-alternatives -s java-6-sun
```

Dependiendo de cual versión de Java se instale es como se instalará Hadoop, debido a las prontas modificaciones a hacer en los ficheros de Hadoop para su integración a Ubuntu. Se puede saber si Java se instaló correctamente consultando a Ubuntu sobre la versión instalada con el siguiente comando:

```
master@master:~$ java -version
```

## Instalación y configuración de Hadoop

Para comenzar con la instalación de Hadoop, primero debe crearse un usuario dedicado. Para esto, primero se crea un grupo de usuarios llamado *hadoop*:

```
master@master:~$ sudo addgroup hadoop
master@master:~$ sudo adduser --ingroup hadoop hduser
```

```
master@master:~$ sudo addgroup hadoop
Añadiendo el grupo 'hadoop' (GID 1001) ...
Hecho.
master@master:~$ sudo adduser --ingroup hadoop hduser
Añadiendo el usuario 'hduser' ...
Añadiendo el nuevo usuario 'hduser' (1001) con grupo 'hadoop' ...
Creando el directorio personal '/home/hduser' ...
Copiando los ficheros desde '/etc/skel' ...
Introduzca la nueva contraseña de UNIX: 
```

*Hduser* es el nombre del usuario dedicado a Hadoop que se utilizará en esta guía, y ha sido agregado al grupo de usuarios *hadoop* en la máquina local.

Hadoop requiere acceso SSH para poder gestionar los nodos, por lo que se genera una llave SSH para el usuario *hduser*. Para esto, se debe iniciar sesión con el usuario *hduser* a través de la terminal:

```
master@master:~$ su - hduser
```

Y luego, generar la llave SSH para *hduser*:

```
hduser@master:~$ ssh-keygen -t rsa -P ""
```

Debe permitirse el acceso SSH a la máquina local para el usuario *hduser*:

```
hduser@master:~$ cat $HOME/.ssh/id_rsa.pub >> $HOME/.ssh/authorized_keys
```

Finalmente, se prueba si SSH se instaló correctamente conectando a la máquina local con el usuario *hduser*

```
hduser@master:~$ ssh localhost
```

```
master@master:~$ sudo addgroup hadoop
Añadiendo el grupo 'hadoop' (GID 1001) ...
Hecho.
master@master:~$ sudo adduser --ingroup hadoop hduser
Añadiendo el usuario 'hduser' ...
Añadiendo el nuevo usuario 'hduser' (1001) con grupo 'hadoop' ...
Creando el directorio personal '/home/hduser' ...
Copiando los ficheros desde '/etc/skel' ...
Introduzca la nueva contraseña de UNIX: 
```

Después de esto, se debe desactivar IPv6 en Ubuntu por su uso de la red. Para esto, debe buscarse el fichero *sysctl.conf* y agregar las siguientes líneas al final del fichero:

```
# disable ipv6
net.ipv6.conf.all.disable_ipv6 = 1
net.ipv6.conf.default.disable_ipv6 = 1
net.ipv6.conf.lo.disable_ipv6 = 1
```

Se puede probar si se desactivo correctamente ingresando el siguiente comando en la terminal:



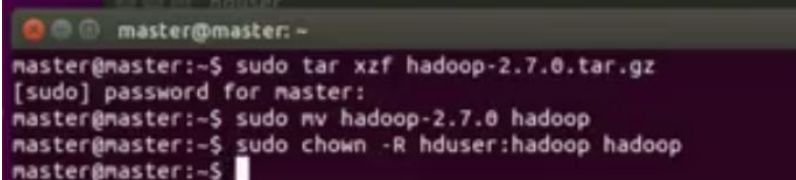
```
hduser@master:~$ cat /proc/sys/net/ipv6/conf/all/disable_ipv6
```

Debería imprimir el valor 1.

## Instalación de Hadoop

Se descarga el archivo de Hadoop a utilizar, y se deja en el directorio donde se quiere instalar Hadoop. En este caso, se utilizó el directorio `/home/master/hadoop`, y se ingresan en la terminal los siguientes comandos:

```
master@master:~$ cd /home/master
master@master:~$ sudo tar xzf hadoop-2.7.0.tar.gz
master@master:~$ sudo mv hadoop-2.7.0 hadoop
master@master:~$ sudo chown -R hduser:hadoop hadoop
```



Debe modificarse el fichero `.bashrc`, ubicado en el directorio de `hduser`:

```
master@master:/home/hduser$ sudo nano .bashrc
```

Y agregar las siguientes líneas al final del archivo (la versión de Java depende de cual versión se tenga instalada en Ubuntu):

```

# Set Hadoop-related environment variables
export HADOOP_HOME=/usr/local/hadoop

# Set JAVA_HOME (we will also configure JAVA_HOME directly for Hadoop later
on)
export JAVA_HOME=/usr/lib/jvm/java-6-sun

# Some convenient aliases and functions for running Hadoop-related commands
unalias fs &> /dev/null
alias fs="hadoop fs"
unalias hls &> /dev/null
alias hls="fs -ls"

# If you have LZO compression enabled in your Hadoop cluster and
# compress job outputs with LZOP (not covered in this tutorial):
# Conveniently inspect an LZOP compressed file from the command
# line; run via:
#
# $ lzohead /hdfs/path/to/lzop/compressed/file.lzo
#
# Requires installed 'lzop' command.
#
lzohead () {
    hadoop fs -cat $1 | lzop -dc | head -1000 | less
}

# Add Hadoop bin/ directory to PATH
export PATH=$PATH:$HADOOP_HOME/bin

```



```

master@master: /home/hduser
GNU nano 2.2.6      Archivo: .bashrc      Modificado

if ! shopt -oq posix; then
  if [ -f /usr/share/bash-completion/bash_completion ]; then
    . /usr/share/bash-completion/bash_completion
  elif [ -f /etc/bash_completion ]; then
    . /etc/bash_completion
  fi
fi

export HADOOP_HOME=/usr/local/hadoop
export JAVA_HOME=/usr/lib/jvm/java-1.7.0-openjdk-amd64

unalias fs &> /dev/null
alias fs="hadoop fs"
unalias hls &> /dev/null
alias hls="fs -ls"

lzohead () {
    hadoop fs -cat $1 | lzop -dc | head -1000 | less
}
Nombre del archivo a escribir: .bashrc
^O Ver ayuda      ^O Formateo DOS  ^O-A Añadir      ^O-B Respalda fich
^O Cancelar      ^O-M Formateo Mac ^O-P Anteponer

```

Debe modificarse el fichero *etc/hadoop-env.sh* ubicado dentro de la carpeta de Hadoop a lo que se indicará con las siguientes líneas, para que se integra con la versión de Java instalado en el sistema (depende de la versión de Java instalado):

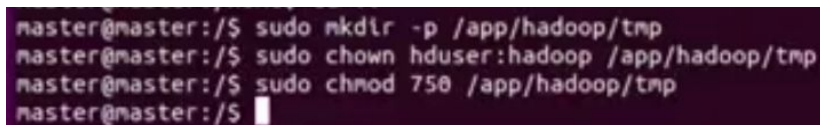
```
# The java implementation to use. Required.  
export JAVA_HOME=/usr/lib/jvm/java-6-sun
```

Se crea el directorio donde se guardarán los archivos de Hadoop a través del sistema HDFS. En este caso, desde el directorio raíz *root* se creará el directorio *app/hadoop/tmp*.

```
master@master:~$ sudo mkdir -p /app/hadoop/tmp  
master@master:~$ sudo chown hduser:hadoop /app/hadoop/tmp
```

Y si se quiere reafirmar la seguridad del directorio:

```
master@master:~$ sudo chmod 750 /app/hadoop/tmp
```

A screenshot of a terminal window with a dark background and light-colored text. It shows four lines of commands being executed: 'naster@master:/S\$ sudo mkdir -p /app/hadoop/tmp', 'naster@master:/S\$ sudo chown hduser:hadoop /app/hadoop/tmp', 'naster@master:/S\$ sudo chmod 750 /app/hadoop/tmp', and 'naster@master:/S\$' followed by a cursor. The first line has a typo 'naster' instead of 'master'.

Luego, se modifican los siguientes ficheros, siempre dentro de los tags `<configuration></configuration>` respectivamente. De la misma forma, estos ficheros se ubican en dentro de la carpeta de Hadoop.

En el fichero *etc/core-site.xml*:

```
<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://localhost: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>
```

En el fichero *etc/mapred.site.xml*

```

<property>
  <name>mapred.job.tracker</name>
  <value>localhost: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>

```

Y en el fichero etc/hdfs-site.xml:

```

<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>

```

## Multinodo y ejecución

Los siguientes pasos corresponden a la implementación de un sistema multinodo, por lo que algunos comandos y ediciones de archivos serán distintos entre las distintas máquinas. Para que Hadoop funcione correctamente, las rutas de la carpeta Hadoop en las máquinas debe ser la misma.

En ambas máquinas, estando ya conectadas dentro de una misma red LAN, debe especificarse cual corresponderá a la máquina maestra y cual(es) serán los esclavos. Para esto, debe modificarse el fichero `/etc/hosts` desde el directorio `root` de los sistemas del clúster.

```

192.168.0.3    master
192.168.0.5    slave

```



La dirección IP depende obviamente de la dirección IP de las máquinas. Como esta guía se realizó en su mayor parte una máquina virtual, la dirección IP será distinta.

El hduser de la máquina maestra debe poder acceder a su propia cuenta de usuario de Hadoop, y a el hduser de la máquina esclava a través de un login SSH sin contraseña:

```
hduser@master:~$ ssh-copy-id -i $HOME/.ssh/id_rsa.pub hduser@slave
```

La sección del comando `hduser@slave` debe corresponderse con el(los) usuario de la(s) máquinas esclavas, así como el nombre de los equipos.

En la máquina maestra, debe crearse un nuevo fichero llamado *masters* ubicado en la carpeta *etc* dentro del directorio de Hadoop, y agregar el nombre de la máquina maestra:

*master* #Este nombre debe corresponderse al nombre de la máquina maestra

En el mismo directorio también debería existir un fichero llamado *slaves*, donde deben existir las siguientes líneas:

*master* #Este nombre debe corresponderse al nombre de la máquina maestra  
*slave* #Este nombre debe corresponderse al nombre de la máquina esclava.

Los ficheros *masters* y *slaves* definen la jerarquía de las máquinas conectadas al clúster. haber puesto el nombre de la máquina maestra en ambos ficheros permite a la máquina maestra funcionar también como esclava. Si se quieren agregar más máquinas como esclavas, simplemente deben agregarse sus nombres al fichero *slaves*.

En todas las máquinas del clúster, deben modificarse los ficheros con las líneas destacadas:

En el fichero *etc/core-site.xml*:

```
<property>
  <name>fs.default.name</name>
  <value>hdfs://master:54310</value> #master debe cambiarse por el nombre
  de la máquina maestra
  <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>
```

En el fichero *etc/mapred-site.xml*:

```
<property>
  <name>mapred.job.tracker</name>
  <value>master: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>
```

En el fichero *etc/hdfs-site.xml*:

```
<property>
  <name>dfs.replication</name>
  <value>2</value> #Este valor especifica la cantidad de máquinas a conectar
  al clúster.
  <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>
```

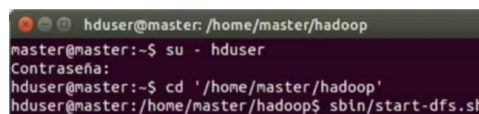
## Resultados

Para poder iniciar Hadoop, primero debe formatearse HDFS. Esto se realiza la primera vez que se inicia el clúster de Hadoop:

```
hduser@master:/home/master/hadoop$ sbin/hadoop namenode -format
```

Y luego, se inicia el clúster con el siguiente comando (para todas las máquinas):

```
hduser@master:/home/master/hadoop$ sbin/start-dfs.sh
```



```
hduser@master:/home/master/hadoop
master@master:~$ su - hduser
Contraseña:
hduser@master:~$ cd '/home/master/hadoop'
hduser@master:/home/master/hadoop$ sbin/start-dfs.sh
```

Como se mencionó anteriormente, la ruta donde se instaló Hadoop debe ser la misma en todas las máquinas. El comando anterior inicia los demonios necesarios de Hadoop en las máquinas del clúster. Para revisar si se inició correctamente, debe ingresarse el siguiente comando que muestra los demonios iniciados en cada máquina.

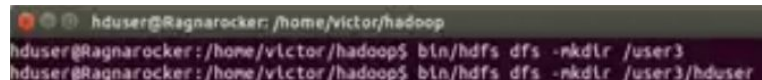
```
hduser@master:/home/master/hadoop$ jps
```

En la máquina maestra, deben iniciarse los demonios NameNode, DataNode y SecondaryNameNode, mientras que en la(s) máquina(s) debe iniciarse el demonio DataNode.

## Usando Hadoop

Para crear directorios en el HDFS que puedan ser accedidos desde las demás máquinas, debe iniciarse sesión con la cuenta de usuario `hduser` de Hadoop en la máquina maestra e ingresar el siguiente comando:

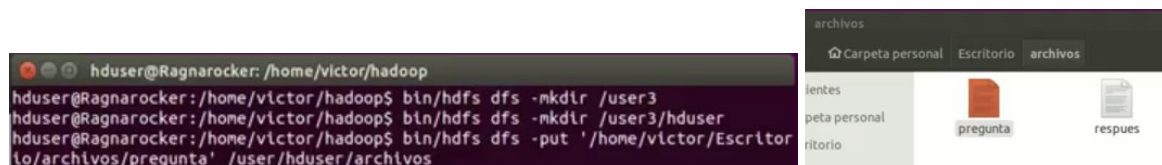
```
hduser@master:/home/master/hadoop$ bin/hdfs dfs -mkdir /directorio
```

A terminal window showing the execution of HDFS commands. The prompt is `hduser@Ragnarocker:/home/victor/hadoop`. The first command is `bin/hdfs dfs -mkdir /user3` and the second is `bin/hdfs dfs -mkdir /user3/hduser`.

```
hduser@Ragnarocker:/home/victor/hadoop$ bin/hdfs dfs -mkdir /user3
hduser@Ragnarocker:/home/victor/hadoop$ bin/hdfs dfs -mkdir /user3/hduser
```

Luego, deben agregarse los ficheros que se quieran trabajar en HDFS al directorio HDFS:

```
hduser@master:/home/master/hadoop$ bin/hdfs dfs -put
'directorio_de_fichero1' 'directorio_donde_se_guardan_los_ficheros_en_HDFS'
```

Two screenshots are shown side-by-side. The left one is a terminal window showing the creation of directories `/user3` and `/user3/hduser`, and the execution of `bin/hdfs dfs -put '/home/victor/Escritorio/archivos/pregunta' /user/hduser/archivos`. The right one is a file manager window showing the 'archivos' folder with files 'pregunta' and 'respuesta'.

```
hduser@Ragnarocker:/home/victor/hadoop$ bin/hdfs dfs -mkdir /user3
hduser@Ragnarocker:/home/victor/hadoop$ bin/hdfs dfs -mkdir /user3/hduser
hduser@Ragnarocker:/home/victor/hadoop$ bin/hdfs dfs -put '/home/victor/Escritorio/archivos/pregunta' /user/hduser/archivos
```

El directorio donde se guardan los ficheros es la carpeta dentro de HDFS donde se quieren guardar, por lo que en este caso corresponde a “`/directorio`” como se mostró en el comando anterior. Si se quieren agregar más ficheros, debe realizarse este mismo comando, cambiando el nombre del fichero y, de ser necesario, el directorio del fichero.

Para ver si se guardan los ficheros, se usa el siguiente comando:

```
hduser@master:/home/master/hadoop$ bin/hdfs dfs -ls 'carpeta_en_HDFS'
```

Hadoop permite la ejecución de proyectos Java en HDFS para trabajar los ficheros. los siguientes pasos comprenden la creación de un archivo `.jar`, que es el que usa Hadoop para trabajar con Java.

Se abre una nueva terminal para poder usar los permisos `sudo` de Ubuntu, y se compila el proyecto `.java` a utilizar. En este caso, se utilizará el programa `WordCount`, que se considera como ejemplo clásico para trabajar con Hadoop:

```
master@master:~/directorio_de_.java$ javac -classpath
/home/master/hadoop/share/hadoop/common/hadoop-common-2.7.0.jar;/home/maste
r/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-client-core-2.7.0.jar;/hom
e/master/hadoop/share/hadoop/common/lib/commons-cli-1.2.jar -d
'directorio_de_.java' *.java
```

El comando anterior importa las bibliotecas necesarias para poder compilar este proyecto .java, debido a las librerías que necesita. Una vez compilado, puede verse que se crearon tres clases: WordCount.class, WordCount\$Map.class, y WordCount\$Reduce.class.



Estas clases se mueven al directorio wordcountclass.

Se ingresa el siguiente comando, el cual crea el archivo .jar necesario para trabajar con los ficheros dentro del HDFS:

```
master@master:~/directorio_de_.java$ jar -cvf wordcount.jar -C
'directorio_donde_estan_las_clases'
```

```
victor@Ragnarocker:~/Escritorio/wordcountarchivos$ jar -cvf wordcountj.jar -C /h
one/victor/Escritorio/wordcountarchivos/wordcountclass .
Picked up JAVA_TOOL_OPTIONS: -javaagent:/usr/share/java/jayatanaag.jar
manifiesto agregado
agregando: WordCount$Map.class(entrada = 1855) (salida = 781)(desinflado 57%)
agregando: WordCount.class(entrada = 1478) (salida = 737)(desinflado 50%)
agregando: WordCount$Reduce.class(entrada = 1627) (salida = 685)(desinflado 57%)
```

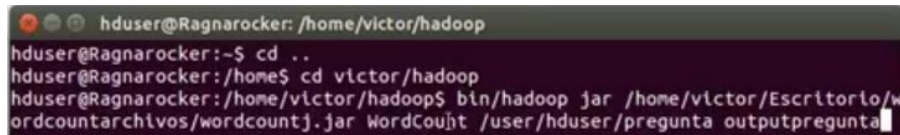
Puede verse que se creó el archivo .jar a utilizar.



Ahora, se inicia sesión con el usuario *hduser*, dirigiéndose al directorio de Hadoop, y se compila el .jar con los ficheros del directorio dentro de HDFS:



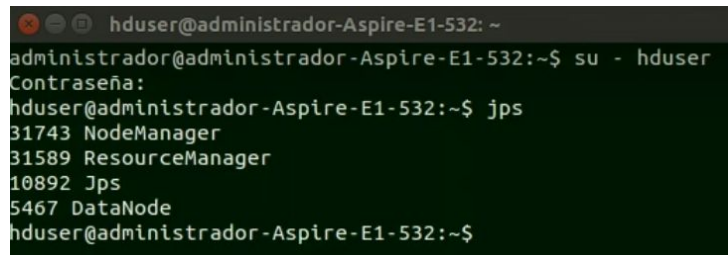
```
hduser@master:~$ /home/master/hadoop bin/hadoop jar  
'Directorio_donde_se_encuentra_el_archivo_.jar' WordCount  
'Fichero_a_trabajar_con_Java' 'Directorio_de_salida_dentro_de_HDFS'
```



```
hduser@Ragnarocker: /home/victor/hadoop  
hduser@Ragnarocker:~$ cd ..  
hduser@Ragnarocker:/home$ cd victor/hadoop  
hduser@Ragnarocker:/home/victor/hadoop$ bin/hadoop jar /home/victor/Escritorio/w  
ordcountarchivos/wordcountj.jar WordCount /user/hduser/pregunta outputpregunta
```

La sección “WordCount” en el comando anterior se utiliza por el nombre del proyecto. Una vez terminado, se puede ver que se creó un directorio de salida, el cual posee el fichero de salida con los resultados de la compilación. Si Hadoop se configuró correctamente, esta salida también podrá accederse desde la máquina esclava.

En la máquina esclava, ingresamos a la cuenta de usuario de Hadoop a través de la terminal.



```
hduser@administrador-Aspire-E1-532: ~  
administrador@administrador-Aspire-E1-532:~$ su - hduser  
Contraseña:  
hduser@administrador-Aspire-E1-532:~$ jps  
31743 NodeManager  
31589 ResourceManager  
10892 Jps  
5467 DataNode  
hduser@administrador-Aspire-E1-532:~$
```

Y para ingresar a Hadoop desde la máquina esclava, debe ingresarse en el cuadro de enlace del navegador la dirección IP de la máquina maestra.

192.168.0.5:50070

En la pestaña *Utilities*, en la sección *Browse the File System*, se ingresa a HDFS, donde puede verse los directorios gestionados por la máquina maestra.

También puede verse el directorio de salida creado anteriormente, desde donde se puede descargar el fichero resultado de la compilación.

Finalmente, se ve que el fichero de salida fue creado y accedido correctamente desde la máquina esclava.

# Browse Directory

/user/hduser

Go!

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxr-xr-x	hduser	supergroup	0 B	9-9-2015 00:05:24	0	0 B	<a href="#">archivos</a>
drwxr-xr-x	hduser	supergroup	0 B	9-9-2015 11:41:13	0	0 B	<a href="#">output</a>
drwxr-xr-x	hduser	supergroup	0 B	9-9-2015 11:49:57	0	0 B	<a href="#">outputpregunta</a>
-rw-r--r--	hduser	supergroup	28.25 KB	9-9-2015 00:07:51	2	128 MB	<a href="#">pregunta</a>
-rw-r--r--	hduser	supergroup	15.48 KB	9-9-2015 00:07:59	2	128 MB	<a href="#">respues</a>
drwxr-xr-x	hduser	supergroup	0 B	9-9-2015 03:57:05	0	0 B	<a href="#">salida</a>
drwxr-xr-x	hduser	supergroup	0 B	9-9-2015 03:19:30	0	0 B	<a href="#">salidas</a>