Hadoop y Map Reduce en AWS.

José Incera, Abril 2019

Introducción

En esta práctica nos familiarizaremos con una de las arquitecturas de cómputo distribuido más populares en la actualidad: El sistema Hadoop. Utilizaremos el paradigma de programación MapReduce.

- En la primera parte, desplegaremos una instancia de Hadoop en un solo nodo en Amazon EC2.
- En la segunda, ejecutaremos algunos programas bajo el paradigma MapRedice en se nodo.
- En la tercera parte, extenderemos nuestro ambiente a un cluster de cuatro nodos.
- En la cuarta y última parte, correremos algunos programas MapReduce en nuestro cluster.

Objetivos

- Reafirmar conceptos básicos de la arquitectura Hadoop y el modelo de programación MapReduce.
- Desplegar un cluster de Hadoop en la nube Amazon EC2

1. Instalación de un nodo Hadoop en AWS EC2

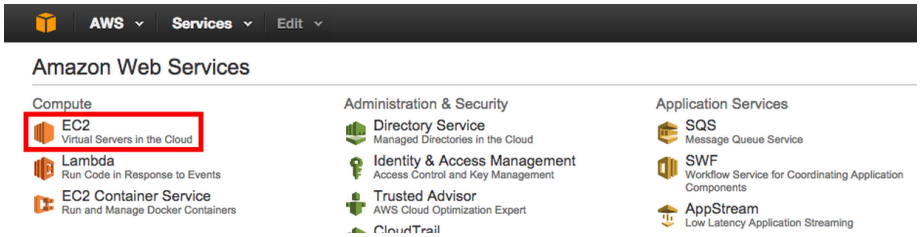
Esta sección está basada en los tutoriales:

- Heleicui: http://www.cs.cityu.edu.hk/~heleicui2/doc/Setup-Hadoop-2.7.3-(single-node)-on-AWS-EC2-Ubuntu-AMI.pdf
- Ouyang: https://blog.insightdatascience.com/spinning-up-a-free-hadoop-cluster-step-by-step-c406d56bae42

1.1 Crear una instancia EC2 en AWS

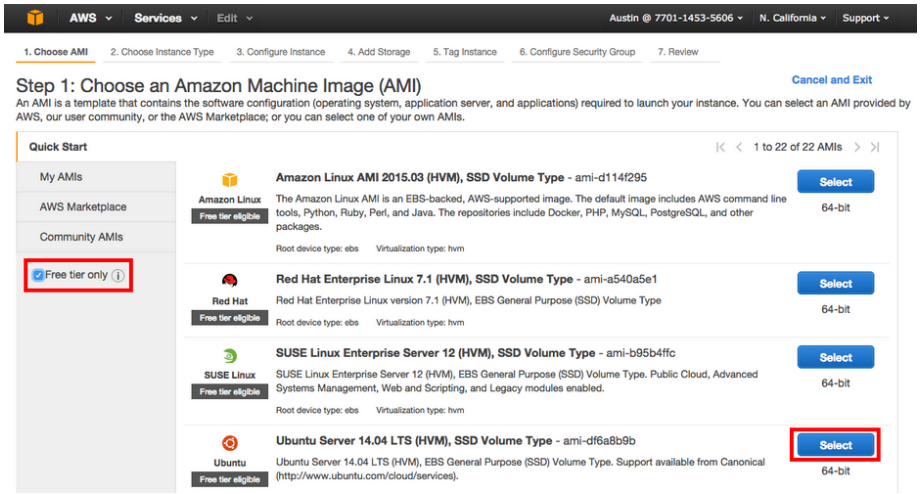
1.- Ingrese a su cuenta AWS. Si aún no tiene una, es un buen momento para crearla en aws.amazon.com Los servicios gratuitos son más que suficientes para nuestro proyecto.

2.-De clic en EC2 (*Elastic Cloud Compute*) y lance una máquina virtual



3.- Seleccione la región (puede ser cualquiera)

4.- Para la imagen de la máquina virtual, seleccione *Ubuntu Server 14.04 LTS (HVM)* y de clic en *Free tier only.*



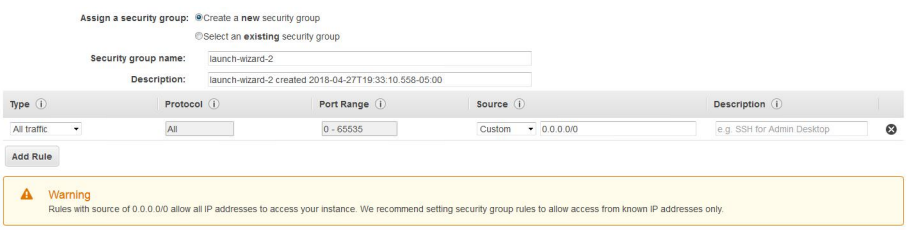
5.- Para el tamaño, `t2.micro` es suficiente. De clic en *Next: Configure Instance Details.*

- Seleccione una sola instancia
- De clic en *Prevention against accidental termination.*

6.- De clic en *Next: Add Storage*. Dejaremos el almacenamiento por default. Para otros casos, aquí puede aumentar el tamaño.

7.- De clic en *Add Tags* y agregue una etiqueta *Key*: `name` y *Value*: `master`

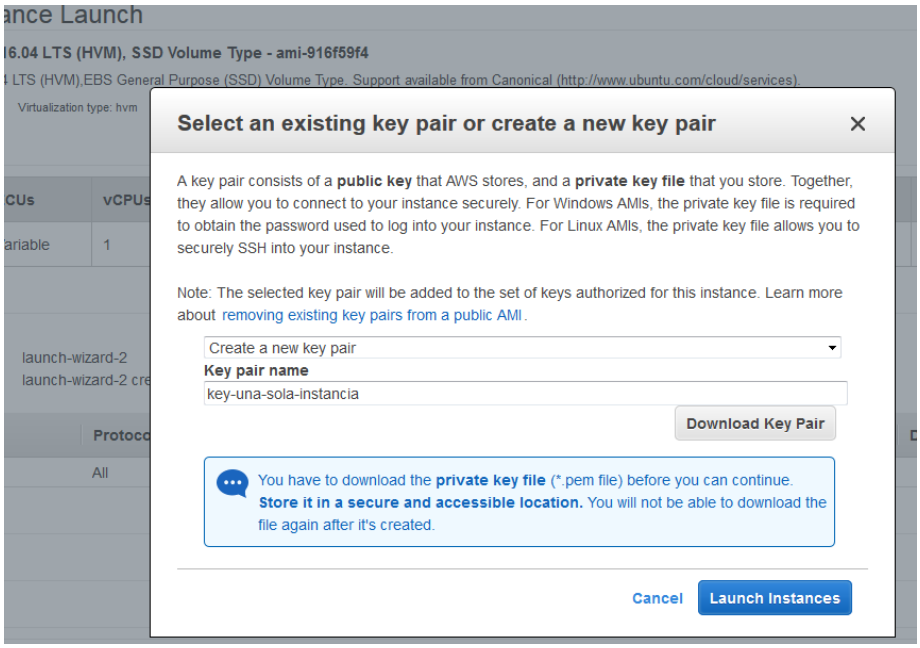
8.- De clic en *Next Configure Security Group*. Por simplicidad, por ahora dejaremos abiertos todos los puertos a todo mundo. Por supuesto, es muy irresponsable trabajar de esta manera tan expuesta en un proyecto real.



9.- De clic en *Review and launch* y si todo está correcto, de clic en *Launch*.

Si es la primera vez que lanza una instancia, se le invitará a generar las llaves PEM necesarias para acceder de forma segura (vía `ssh`) a su instancia.

Seleccione *Create a new key pair*, asígnele un nombre y guárdela en una carpeta.



CUIDADO: Si pierde el archivo que guardó, perderá permanentemente su acceso a la máquina virtual.

De clic en *Launch instances*

10.- De clic en *View Instances*. Aparece una pantalla con las características de la o las instancias que haya lanzado. Selecciónela para ver características con más detalle.

Una muy importante, es el nombre público (*Public DNS*) que se le asignó pues lo estaremos usando intensamente en el resto del tutorial. De clic en el icono con las dos carpetas y guarde el nombre en algún archivo que pueda acceder fácilmente.

En los siguientes comandos, cambie <su_public_DNS> por el nombre que acaba de copiar y grabar.

1.2 Acceder a la instancia

1.2.1.- Linux

Para poder acceder a la instancia con `ssh` en Linux, el archivo donde está la llave PEM debe tener permisos de lectura y escritura únicamente para el dueño.

En los siguientes comandos, cambie <su_archivo_PEM> por el archivo donde almacenó las llaves en el paso **9** anterior.

```
> sudo chmod 600 <su_archivo_PEM>
> ssh -i <su_archivo_PEM> ubuntu@<su_public_DNS>

ubuntu@ip-<ip-privada->$
```

Listo ya puede ingresar a la VM dando clic en `Connect`.

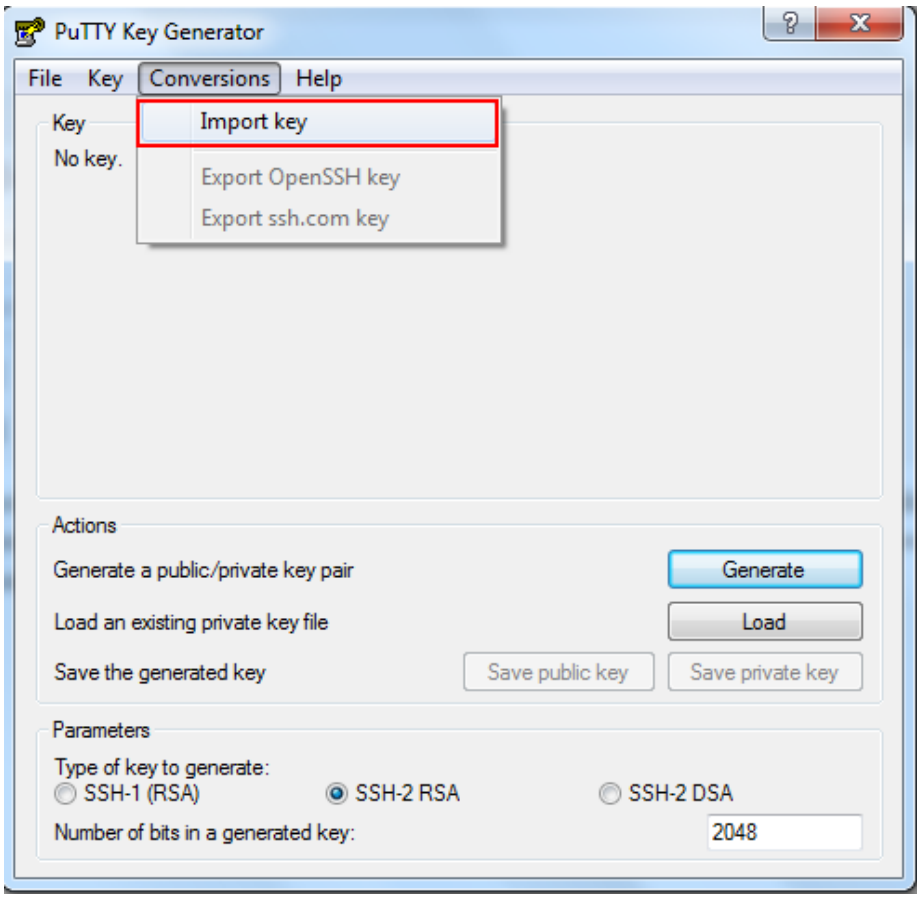
Continúe en la sección **1.3.**

1.2.2.-Windows

Acceder a la VM desde Windows es un poco más complicado porque Windows no tiene implementado el comando `ssh` y la aplicación más popular para hacer conexiones seguras `PuTTY`, no acepta el formato de la llave PEM para hacer la conexión. Por ello, PuTTY también provee una aplicación auxiliar `PuTTYGen`, para generar las llaves compatibles con PuTTY.

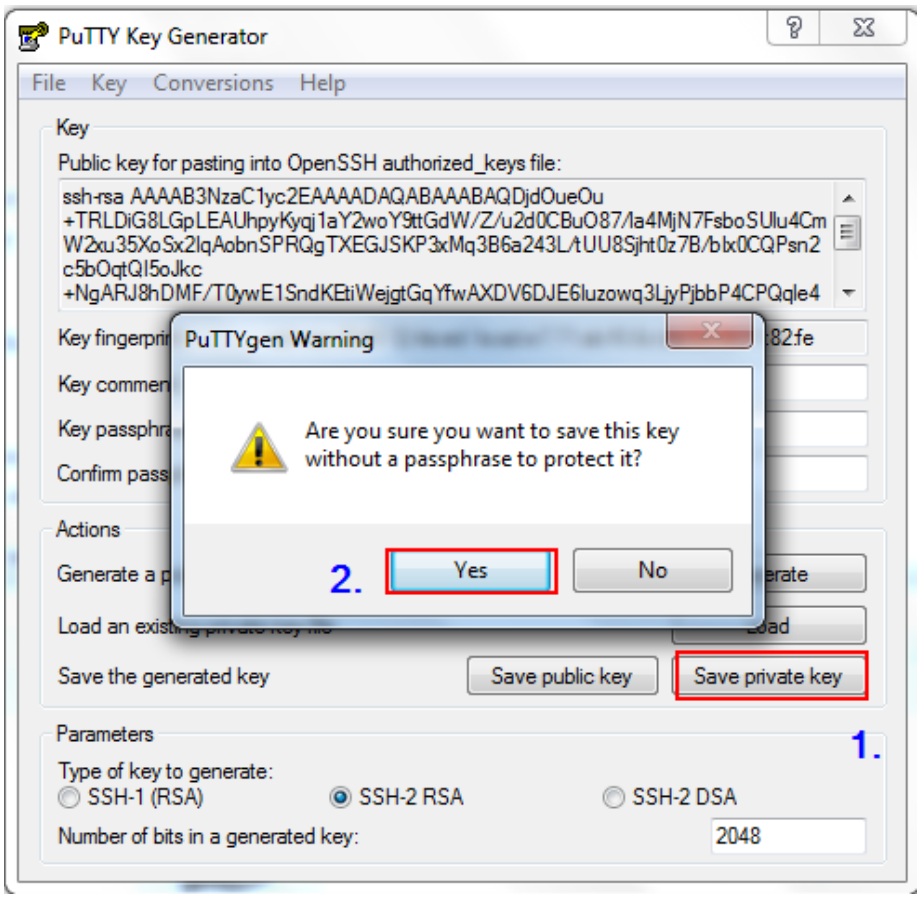
1.- Genere la llave privada.

Lance `PuTTYGen`, de clic en `Conversions` e Importe el archivo con las llaves PEM que se crearon al lanzar la instancia de nuestra VM.



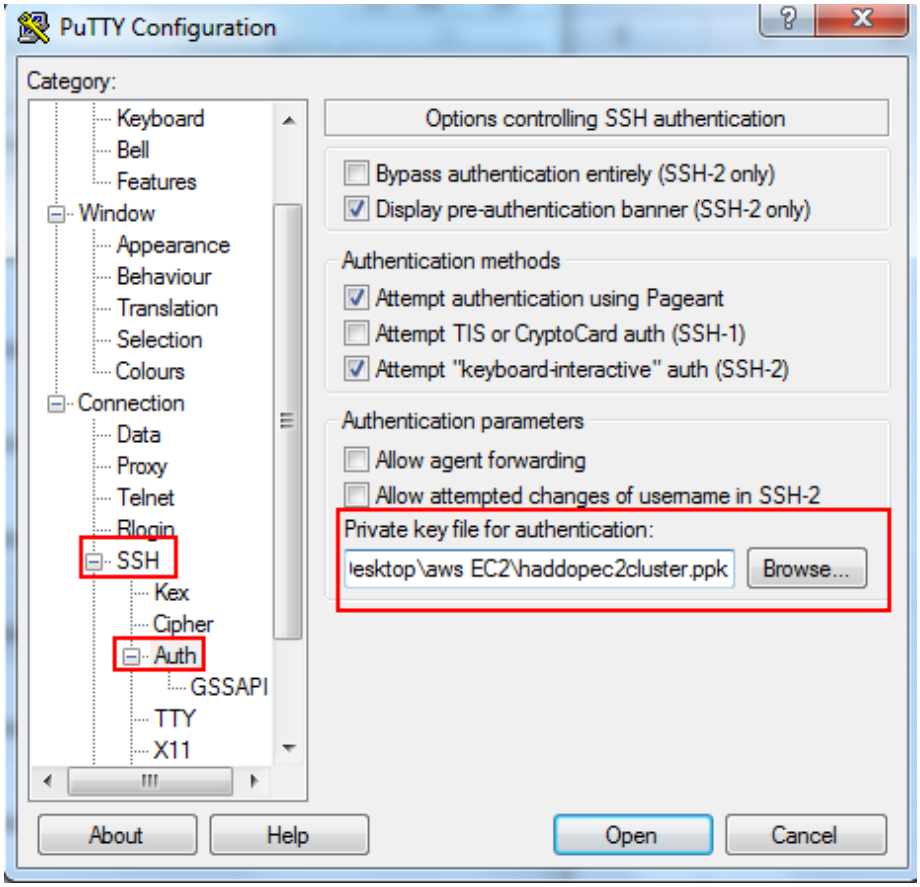
Cargue las llaves, elija la *passphrase* para proteger sus llaves, o deje en blanco esos campos si lo prefiere.

De clic en `Save private key` y guarde la llave *.ppk* en un archivo.

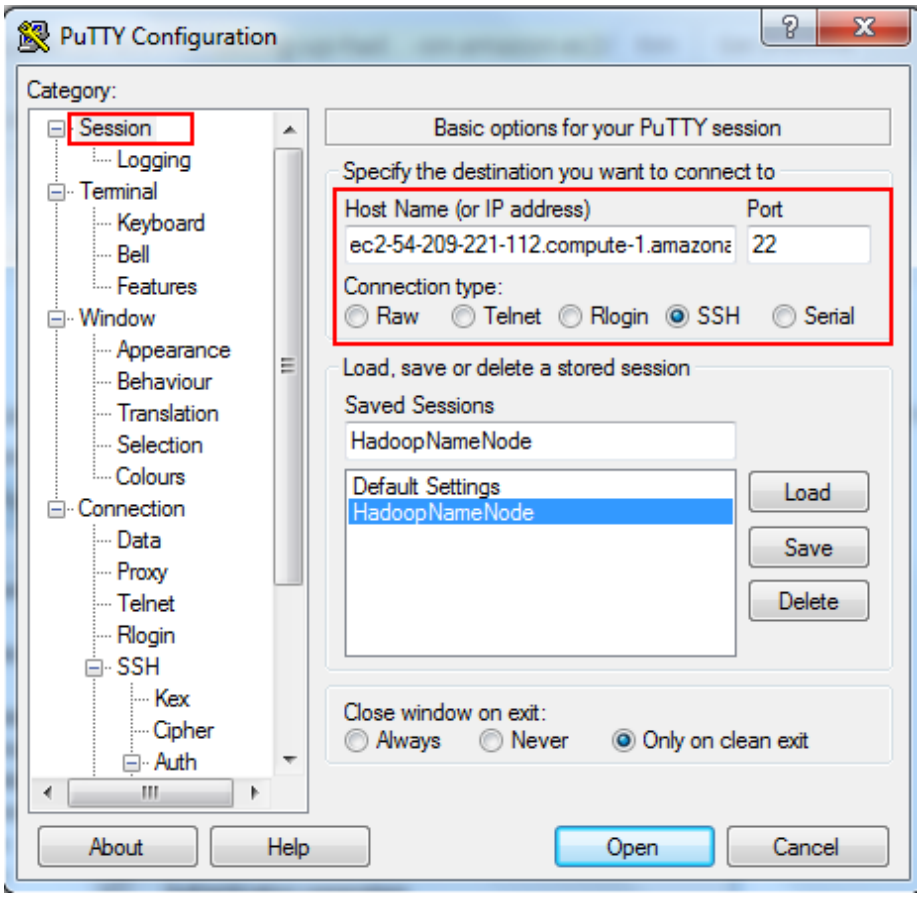


2.- Conexión a la instancia EC2

Lance la aplicación `PuTTY`, de clic en `SSH/Auth` y cargue la llave privada que acaba de guardar.



De clic en `Session` y ponga <su_public_DNS> en la ventana `Host Name`. Para no repetir este paso, puede guardar esta configuración en la sección `Saved Sessions`.



De clic en `Open`. Si despliega un mensaje indicando que la llave no está almacenada, de clic en `Yes`. Se le solicitará un nombre de usuario, ingrese **ubuntu.**

1.3. Instalar Hadoop

Para instalar y ejecutar Hadoop se requiere de Java, que no está instalado en las instancias de AWS. Empecemos por instalar java.

1.- Para instalar java 8 (puede utilizar una versión anterior) desde el repositorio de Oracle, ejecute los siguientes comandos:

```
    > sudo apt-get install python-software-properties
    > sudo add-apt-repository ppa:webupd8team/java
    > sudo apt-get update
    > sudo apt-get install oracle-java8-installer
```

2.- Para comprobar que Java se instaló correctamente, ejecute el siguiente comando:

```
    > java -version

    java version "1.8.0_171"
    Java(TM) SE Runtime Environment (build 1.8.0_171-b11)
    Java HotSpot(TM) 64-Bit Server VM (build 25.171-b11,
mixed mode)
```

3.- Descargue y descomprima la última versión estable de Hadoop de los espejos de Apache. (Para ver cuál es la última versión estable, consulte esta liga.

```
    > wget http://www-
us.apache.org/dist/hadoop/common/hadoop-2.7.6/hadoop-
2.7.6.tar.gz
    > sudo tar xzf hadoop-2.7.6.tar.gz -C /usr/local
    > sudo mv /usr/local/hadoop-* /usr/local/hadoop
```

4.- Variables de ambiente

Se deben agregar una serie de variables de ambiente de Java y de Hadoop. Edite el archivo `.bashrc` e inserte las siguientes líneas:

```
> vim ~/.bashrc

    export JAVA_HOME=/usr/lib/jvm/java-8-oracle
    export HADOOP_HOME=/usr/local/hadoop
    export
HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native

    PATH=$PATH:$JAVA_HOME/bin:$HADOOP_HOME/bin:
```

Guarde el archivo modificado. Para que los cambios tengan efecto en la sesión actual, ejecute el siguiente comando:

```
> source ~/.bashrc
```

5.- Acceso sin contraseña En Hadoop, es un requisito poder acceder al ambiente via `SSH` sin contraseñas. Para ello, se generará un par de llaves sin contraseña y la llave pública se almacenará en el archivo de llaves autorizadas:

```
> ssh-keygen -t rsa -P ''
> cat $HOME/.ssh/id_rsa.pub >> $HOME/.ssh/authorized_keys
```

Ahora debemos modificar el archivo de configuración de SSH. Hay que buscar la línea `PasswordAuthentication no` y cambiar el `no` por `yes`.

- Entre a configurar el archivo: `$ sudo vim /etc/ssh/sshd_config`
- Busque la frase así: `/PasswordAuth`
- Colóquese debajo de la letra 'n' y sustituya la palabra así: `cw yes ^ESC`

Reinicie el servicio ssh para que las actualizaciones se incluyan

```
> sudo service ssh restart
```

Para verificar que puede entrar automáticamente (sin contraseña) acceda a localhost.

```
> ssh localhost

# Este es un nuevo shell. Felicidades
# Ahora salimos y regresamos a la sesión anterior:
> exit
```

6.- Configuración de Hadoop

Para que Hadoop funcione adecuadamente, se deben actualizar varios archivos de configuración. Estos se encuentran en `/usr/local/hadoop/etc/haddop`. Todos los archivos pertenecen a `root`, por lo que es necesario invocar el comando de edición como `sudo vim`.

core-site.xml

Copie y pegue las siguientes líneas:

```
<property>
    <name>hadoop.tmp.dir</name>
    <value>/home/ubuntu/hadooptmp/hadoop-${user.name}
</value>
    <description>A base for other temporary directories.
</description>
</property>
<property>
    <name>fs.defaultFS</name>
    <value>hdfs://localhost:9000</value>
</property>
```

hadoop-env-sh

Debemos remplazar la ubicación del directorio home de Java. De `${JAVA_HOME}` a:

```
export JAVA_HOME=/usr/lib/jvm/java-8-oracle
```

mapred-site.xml

Genere el archivo a partir de su plantilla:

```
> sudo cp mapred-site.xml.template mapred-site.xml
```

Abra el nuevo archivo, copie y pegue las siguientes líneas:

```
<property>
    <name>mapred.job.tracker</name>
    <value>localhost:9001</value>
</property>
```

yarn-site.xml

```
<property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
</property>
<property>
    <name>yarn.nodemanager.aux-
services.mapreduce.shuffle.class</name>
    <value>org.apache.hadoop.mapred.ShuffleHandler</value>
</property>
<property>
    <name>yarn.resourcemanager.hostname</name>
    <value>localhost</value>
</property>
```

hdfs-site.xml

En este archivo se especifica el factor de replicación y la ubicación de las carpetas para los datos del NameNode y del DataNode.

Copie y pegue las siguientes líneas

```
<property>
    <name>dfs.replication</name>
    <value>1</value>
</property>
<property>
    <name>dfs.namenode.name.dir</name>

<value>file:///usr/local/hadoop/hadoop_data/hdfs/namenode<
/value>
</property>
<property>
    <name>dfs.datanode.data.dir</name>

<value>file:///usr/local/hadoop/hadoop_data/hdfs/datanode<
/value>
</property>
```

Los directorios especificados no existen. Debemos crearlos.

```
> sudo mkdir -p $HADOOP_HOME/hadoop_data/hdfs/datanode
> sudo mkdir -p $HADOOP_HOME/hadoop_data/hdfs/namenode
```

Ahora cambiaremos la propiedad de todos los archivos en $HADOOP_HOME al usuario `ubuntu`.

```
> sudo chown -R ubuntu $HADOOP_HOME
```

Antes de poder trabajar con el sistema de archivos HDFS, debemos formatearlo.

```
> hdfs namenode -format
```

Estamos listos para iniciar HDFS y YARN:

```
    > $HADOOP_HOME/sbin/start-dfs.sh
    > $HADOOP_HOME/sbin/start-yarn.sh
```

Si se le indica *The authenticity of host 'Some Node' can't be established. Are you sure you want to continue connecting (yes/no)?* simplemente responda `yes`.

Puede checar que el nodo está levantado desde una interfaz de su navegador en el URL su_public_DNS:50070.

También puede verificar que todos los procesos de hadoop se están ejecutando en la máquina virtual de Java:

```
> jps

19553 DataNode
20338 Jps
19734 SecondaryNameNode
20025 NodeManager
19884 ResourceManager
19407 NameNode
```

7.- Cambiar hostname Por último, para no confundirnos más adelante, cambiemos el nombre del host por el de `NameNode`:

```
> sudo hostnamectl set-hostname 'NameNode'
> hostname
NameNode
```

2. El proyecto en MapReduce

En un escenario electoral hipotético, se realizaron encuestas de salida para conocer las preferencias de los electores, así como algunos datos demográficos. De estas encuestas se generó el archivo *votacion.csv* el cual contiene cuatro campos:

1. Hora.- Número entero en el rango [8:17], registra la hora en que se aplicó la encuesta al elector
2. Género.- H = Hombre, M = Mujer, se trata del género del elector
3. Distrito.- Un código que representa el distrito electoral en el que se aplicó la encuesta
4. Candidato.- Número entero en el rango [1:5], representa cada uno de los cinco candidatos que se postularon.

Aunque el archivo *votacion.csv* es muy pequeño, en esta práctica se almacenará en una instancia de Hadoop en su máquina virtual y se realizarán algunos análisis básicos con el modelo MapReduce utilizando guiones (scripts) en Python.

Esta práctica también permitirá experimentar con algunos comandos básicos de Unix/Linux.

Preparación de datos

En esta práctica trabajaremos exclusivamente a través de la interfaz de la línea de comandos (CLI). Si no lo ha hecho, conéctese a la máquina virtual de su ambiente Hadoop.

```
hdp>
```

Para poder ejecutar las prácticas debemos crear un directorio en HDFS para nuestro usuario.

```
hdp> hdfs dfs -mkdir -p /user/ubuntu
```

1.- En su directorio `$HOME` cree una carpeta Pr5 con dos subcarpetas: `code` y `data`.

```
hdp> cd
hdp> mkdir -p Pr5/code
hdp> mkdir -p Pr5/data
hdp> ls -l Pr5
total 8
drxwr-xr-x 2 ubuntu ubuntu 4096 may 21 18:40 code
drxwr-xr-x 2 ubuntu ubuntu 4096 may 21 18:40 data
```

2.- Descargue los archivos *EjMapper.py* y *EjReducer.py* y guárdelos en la carpeta `code` creada anteriormente. De la misma forma, descargue el archivo *votacion.csv* y guárdelo en la carpeta `data`.

```
hdp> cd Pr5/code
hdp> wget https://raw.githubusercontent.com/jincera/Test-
Repo/master/EjMapper.py
hdp> wget https://raw.githubusercontent.com/jincera/Test-
Repo/master/EjReducer.py
hdp> ls
EjMapper.py  EjReducer.py

hdp> cd ../data
hdp> wget https://raw.githubusercontent.com/jincera/Test-
Repo/master/votacion.csv
hdp> ls
votacion.csv
```

3.- Ubique el archivo *votacion.csv* y despliegue las primeras líneas.

```
hdp> head -3 votacion.csv
12,M,1048,CAND5
15,H,7932,CAND1
13,H,7373,CAND4
```

Como puede observar, se trata de un archivo csv en el que los campos están separados por comas y no tiene encabezado.

Es una buena práctica probar los scripts de Map y Reduce con un conjunto pequeño de datos y, dentro de lo posible, paso a paso desde la línea de comandos, aprovechando los *pipes* de Linux.

4.- Prepare un archivo de prueba con los primeros 100 registros

```
hdp> head -100 votacion.csv > vottst.csv
```

Los archivos se encuentran en el sistema de archivos local. Hay que enviarlos a HDFS, el sistema de archivos distribuido de Hadoop:

```
hdp> hdfs dfs -put votacion.csv
hdp> hdfs dfs -ls
...
-rw-r--r-- 1 ubuntu supergroup 1591000 2017-05-21 18:59
votacion.csv
...
```

El comando *hdfs dfs* (o el equivalente *hadoop fs* en la versión anterior) indica a Linux que se introducirá una directiva para el sistema de archivos HDFS. Los argumentos siguientes son la directiva y posibles parámetros.

Programación de scripts

En esta sesión se desarrollarán los scripts con el lenguaje de programación Python. Al ser un lenguaje interpretado, será muy sencillo verificar el comportamiento de los programas map y reduce con pipes de Linux.

Empezaremos por calcular las preferencias electorales para cada uno de los candidatos.

Los archivos *EjMapper.py* y *EjReducer.py* en la carpeta `code`, son ejemplos de un código mapper y de un reducer, respectivamente. El primero lee registros desde la entrada estándar (*stdin*, típicamente el teclado), selecciona dos campos y los imprime en la salida estándar (*stdout*, típicamente la pantalla). Estos dos campos son la tupla < *key,value* > que el reducer tomará para continuar con el procesamiento.

Haga una copia de los archivos `EjMapper.py` y `EjReducer.py` como respaldo en caso de que algo salga mal durante la ejecución de la práctica

```
hdp> cp EjMapper.py EjMapper.py.bak
hdp> cp EjReducer.py EjReducer.py.bak
```

5.- Edite el archivo *EjMapper.py*. Revise el código y modifique la última línea para que se imprima en stdout la columna correspondiente al candidato (key) y un "1" (value). El código del reducer simplemente sumará estas instancias.

6.- Posiciónese en la carpeta *code* y revise los permisos de los archivos *EjMapper.py* y *EjReducer.py*

```
hdp> cd ~Pr5/code
hdp> ls -l
total 8
-rw-r--r-- 1 ubuntu ubuntu 422 may 21 18:44 EjMapper.py
-rw-r--r-- 1 ubuntu ubuntu 819 may 21 18:45 EjReducer.py
```

Para Linux estos archivos no contienen código ejecutable; sólo tienen permisos de lectura y escritura (rw-). Modifique los permisos para que también puedan ser ejecutados:

```
hdp> chmod 764 *py
hdp>ls -l
total 8
-rwxrw-r- 1 ubuntu ubuntuc 422 may 21 18:44 EjMapper.py
-rwxrw-r- 1 ubuntu ubuntu 819 may 21 18:45 EjReducer.py
```

El comando anterior otorga permisos de lectura, escritura y ejecución (7) al dueño, lectura y escritura (6) a los miembros del grupo y sólo lectura (4) a los demás usuarios.

7.- Con ayuda del encadenamiento de comandos (pipelining) en Linux, verifique que el código parece funcionar correctamente.

```
hdp> cat ../data/vottst.csv |./EjMapper.py
...
CAND5   1
CAND4   1
CAND4   1
```

El comando *cat* lee y despliega en pantalla el archivo. El "pipe" (|) toma esa salida y la pasa al siguiente comando, nuestro script *EjMapper.py*, como su propia entrada estándar.

8.- Abra el archivo *EjReducer.py* y analice su contenido.

Este archivo tiene un acumulador para contar el número de ocurrencias de un operador. Dado que a un proceso *Reduce* llegan los datos ordenados, solo hay que incrementar el acumulador mientras el campo operador (key) no cambie. En principio, este código no debe ser modificado.

9.- Nuevamente usaremos los pipes de Linux, para verificar que el Reducer parece hacer su función. El comando *sort* en la siguiente instrucción ordena la salida de nuestro mapper, imitando la operación *shuffle* de MapReduce en Hadoop. La salida del sort se toma como entrada para el script *EjReducer.py*.

```
hdp>  ../data/vottst.csv | ./EjMapper.py | sort |
./EjReducer.py
CAND1   8
CAND2   5
CAND3   17
CAND4   48
CAND5   22
```

Ejecución en Hadoop

Ahora que todo parece funcionar correctamente, se puede enviar el código para ser ejecutado en Hadoop. Dado que los programas están escritos en Python, se requiere de la API *hadoop streaming*, la cual permite lanzar tareas MapReduce escritas en prácticamente cualquier lenguaje capaz de recibir datos de la entrada estándar y de escribir resultados en la salida estándar.

10.- Desde la terminal, ejecute el siguiente comando:

```

```

```
hdp> hadoop jar
/usr/local/hadoop/share/hadoop/tools/lib/hadoop-streaming-
2.7.6.jar -input votacion.csv -output OpElec -mapper
EjMapper.py -reducer EjReducer.py

hdp> hadoop dfs -ls OpElec

Found 2 items
-rw-r--r--   1 ubuntu supergroup          0 2018-04-28
23:49 05/_SUCCESS
-rw-r--r--   1 ubuntu supergroup         64 2018-04-28
23:49 05/part-00000

hdp> hdfs dfs -cat 05/part-00000

CAND1   10056
CAND2   9884
CAND3   15051
CAND4   40018
CAND5   24991
```

Los argumentos de la instrucción anterior son:

- input: El archivo de donde se leerán los datos que se envían a los procesos Map
- output: El directorio donde se almacenan los resultados (o los mensajes de error)
- mapper, reducer: Los scripts con los códigos para los procesos Map y Reduce

Como vamos a estar utilizando frecuentemente esta API, quizás le gustaría crear una variable de ambiente con la ruta del archivo jar para simplificar la escritura del comando:

```
hdp> export
STRJAR=/usr/local/hadoop/share/hadoop/tools/lib/hadoop-
streaming-2.7.6.jar
```

De esta manera, el comando anterior se invocaría así:

```
hdp> hadoop jar $STRJAR -input votacion.csv ...
```

Revise brevemente la salida, compruebe que no hubo errores, identifique cuántas tareas map y reduce se dispararon.

11.- Verifique que el resultado se generó correctamente:

```
hdp> hadoop fs -cat OpcionElectoral/part-00000
CAND1   10056
CAND2   9884
CAND3   15051
CAND4   40018
CAND5   24991
```

Así se nombran los archivos de salida en la primera versión de MapReduce. En la versión más reciente se nombran part-[m|r]-xxxxx para indicar si el archivo se generó a la salida de un Mapper (m) o de un Reducer (r). El número (xxxxx) es un identificador único en esa carpeta para distinguir entre los resultados de (potencialmente) muchas tareas.

12.- Ahora lance el job con la API de Hadoop streaming y el archivo votación. **Aprovecharemos para mostrar cómo se invoca la ejecución de varias tareas Reducer, lo que puede ser útil para procesar grandes volúmenes de datos**.

```
hdp> hadoop jar $STRJAR -input votacion.csv -output
OpcElect2
-mapper miMapper.R -reducer EjReducer.py -numReduceTasks 2
```

Observe que estamos especificando un nuevo directorio de salida. Hadoop no permite reescribir archivos.

Verifique que obtuvo los mismos resultados, solo que en dos archivos de salida:

```
hdp> hdfs dfs -ls OpcionElectoral2
-rw-r--r--  1 root hdfs   0 2017-05-21  22:48
OpcionElectoral2/_SUCCESS
-rw-r--r--  1 root hdfs   0 2017-05-21  22:48
OpcionElectoral2/part-00000
-rw-r--r--  1 root hdfs   0 2017-05-21  22:48
OpcionElectoral2/part-00001

hdp> hdfs dfs -cat OpcionElectoral2/part-00000
CAND1  10056
CAND3  15051
CAND5  24991

hdp> hdfs dfs -cat OpcionElectoral2/part-00001
CAND2  9884
CAND4  40018
```

El ambiente decidió por sí mismo cómo distribuir las llaves entre los dos Reducers. De lo que podemos tener certeza, es que todos los registros con la misma llave, llegaron al mismo Reducer.