**Consigna 1**

En la consola ejecutar el comando

sudo apt install openjdk-8-jdk

ingresar la contaseña para dar permisos de administrador. Luego ejecutar los comandos:

cd /usr/lid/jvm/

ls

Si dentro de ese directorio se puede ver el archivo java-8-openjdk-amd64 jdk-17 , la instalación fue correcta.

Luego volver con el comando cd al directorio raíz, ejecutar sudo nano .bashrc , e ir hacia el final del archivo en la consola.

En un navegador entrar al sitio codewithajun.medium.com , y seleccionar la opción “Install Hadoop on Ubuntu operating system”, copiar los comando de exportación, y pegarlos con Ctrl+Shift+v en la terminal donde se abrió el archivo .bashrc. Ahora hacer Ctrl+o para guardar el archivo y Ctrl+x para salir del archivo. Ejecutar en consola sudo apt-get install ssh

En el navegador buscar “apache hadoop download”, acceder al sitio hadoop.apache.org/releases.html . Seleccionar la opción “binary” y clickear en la dirección <https://dlcdn.apache.org/hadoop/common/hadoop-3.2.3/hadoop-3.2.3.tar.gz>

Una vez descargado el archivo en la consola (en el directorio raíz) ejecutar:

tar –zxvf ~/Downloads/hadoop-3.2.3.tar.gz

Luego de la extracción de data ejecutar:

cd hadoop-3.2.3/  
cd etc/hadoop/  
ls

Realizar cambios en archivos de ese directorio:

* Ejecutar sudo nano hadoop.env.sh. Dentro del archivo descomentar la línea JAVA\_HOME=/usr/java/testing hdfs dfs –ls y cambiarla por JAVA\_HOME=/usr/lib/jvm/java-8-openjdk-amd64 . Ahora Ctrl+o y enter para guardar, y Ctrl+x para salir.
* Ejecutar sudo nano core-site.xml . Desde el sitio codewitharjun.medium.com abierto anteriormente, copiar las líneas en formato xml correspondientes a este archivo. Volver a la consola con el archivo abierto y reemplazar las últimas líneas (no comentadas) con las copiadas. Ahora Ctrl+o y enter para guardar, y Ctrl+x para salir.
* Ejecutar sudo nano hdfs-site.xml. Desde el sitio codewitharjun.medium.com abierto anteriormente, copiar las líneas en formato xml correspondientes a este archivo. Volver a la consola con el archivo abierto y reemplazar las últimas líneas (no comentadas) con las copiadas. Ahora Ctrl+o y enter para guardar, y Ctrl+x para salir.
* Ejecutar sudo nano mapred-site.xml. Desde el sitio codewitharjun.medium.com abierto anteriormente, copiar las líneas en formato xml correspondientes a este archivo. Volver a la consola con el archivo abierto y reemplazar las últimas líneas (no comentadas) con las copiadas. Ahora Ctrl+o y enter para guardar, y Ctrl+x para salir.
* Ejecutar sudo nano yarn-site.xml. Desde el sitio codewitharjun.medium.com abierto anteriormente, copiar las líneas en formato xml correspondientes a este archivo. Volver a la consola con el archivo abierto y reemplazar las últimas líneas (configuration) con las copiadas. Ahora Ctrl+o y enter para guardar, y Ctrl+x para salir.

Una vez realizados los cambios ahora ejecutar

ssh localhost   
ssh-keygen -t rsa -P '' -f ~/.ssh/id\_rsa   
cat ~/.ssh/id\_rsa.pub >> ~/.ssh/authorized\_keys

(Sobreescribir si quí lo pidiese.)

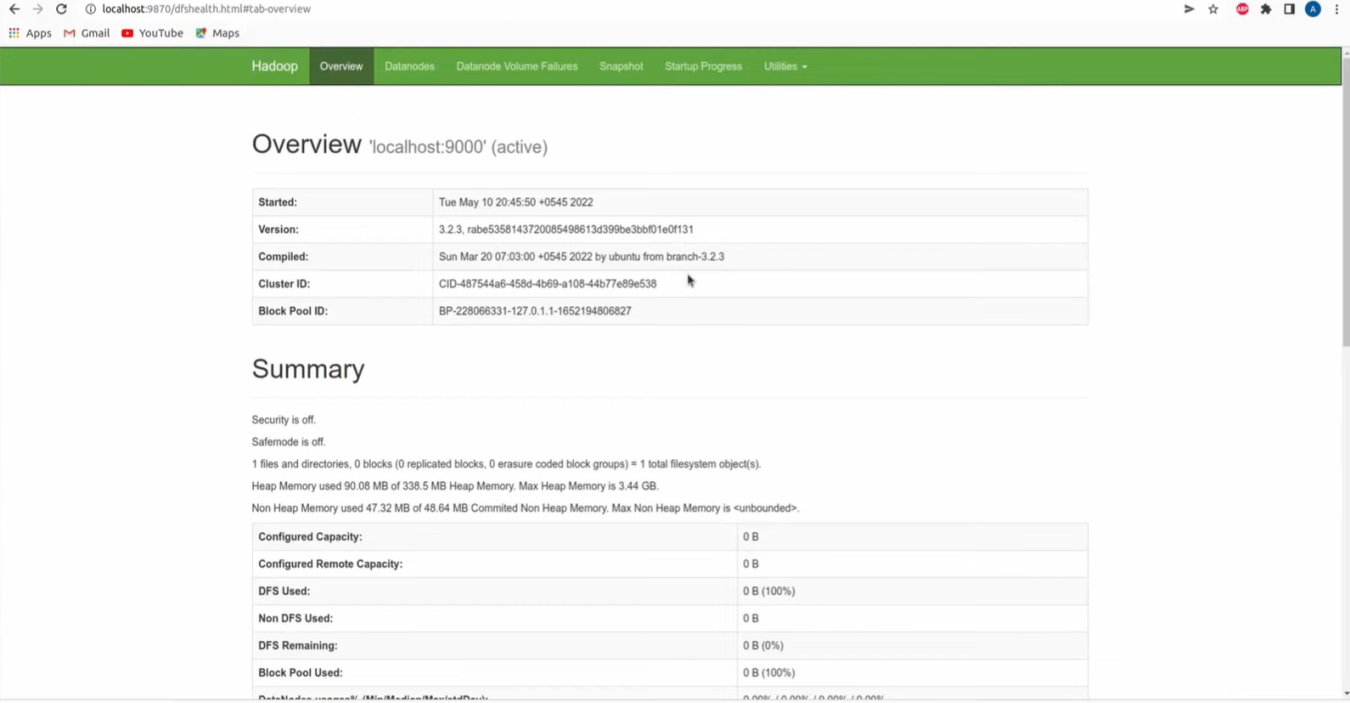
chmod 0600 ~/.ssh/authorized\_keys   
hadoop-3.2.3/bin/hdfs namenode -format

Seleccionar Yes al localhost=null. Luego ejecutar:

export PDSH\_RCMD\_TYPE=ssh

Por último iniciar el programa con start-all.sh

Se puede comprobar que se está ejecutando correctamente en un navegador ingrasando al sitio localhost:9870.



Ejecutar hadoop fs -mkdir /user para crear un nuevo usuario hadoop fs -mkdir /user/code.witharjun.

Ejecutar

touch demo.csv  
hadoop fs -put demo.csv /user/code.witharjun.

**Consigna 2**

Para configurar Hadoop en Google Colaboratory, sigue estos pasos clave:

**Instalar Java 8**

Es necesario para ejecutar Hadoop.

Instalación de Java 8 para una mejor compatibilidad con Hadoop

#Installing java 8

!apt-get install openjdk-8-jdk-headless -qq > /dev/null

# -q, quiet level 2: no output except for errors

#> /dev/null on the end of any command where you want to redirect all the stdout into nothingness

Cambiar la versión predeterminada de Java

#Switching java version to use as default (choose option 2)

!update-alternatives --config java

#Switching javac version to use as default (choose option 2)  
!update-alternatives --config javac

#Switching jps version to use as default (choose option 2)  
!update-alternatives --config jps

#Checking Java default version  
!java –version

Creación de variables de entorno relacionadas con Java

JAVA\_HOME es una variable de entorno del sistema operativo que apunta a la ubicación del sistema de archivos donde se instaló el JDK o JRE.

#Finding the default Java path  
!readlink -f /usr/bin/java | sed "s:bin/java::"

#Importing os module  
import os  
#Creating environment variables  
os.environ["JAVA\_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64"  
os.environ["JRE\_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64/jre"  
os.environ["PATH"] += ":JAVAHOME/bin:JRE\_HOME/bin:HADOOPHOME/bin:HADOOP\_HOME/sbin"

**Instalar SSHD**

Permite la comunicación segura con Hadoop.

Se requiere definir un medio para que el nodo maestro acceda de forma remota a cada nodo de nuestro clúster. Hadoop utiliza frases de contraseña SSH para la comunicación entre los nodos. SSH es un protocolo de red criptográfico para operar servicios de red de forma segura en una red no segura. SSH utiliza criptografía de clave pública estándar para crear un par de claves para la verificación del usuario: una pública y una privada.

#It is good practice to purge before installation  
!apt-get purge openssh-server -qq

#Installing openssh-server  
!apt-get install openssh-server -qq > /dev/null

#Starting the server  
!service ssh start

El número de puerto para SSH es 22 por defecto

!grep Port /etc/ssh/sshd\_config

El modo pseudodistribuido es un caso especial del modo totalmente distribuido, en el que el único host es localhost (nuestra máquina). Debemos asegurarnos de que para acceder a localhost e iniciar sesión no necesitamos ingresar una contraseña. Por lo tanto, SSH debe configurarse para permitir el inicio de sesión sin contraseña para el usuario de Hadoop. La forma más sencilla de lograr esto es generar un par de claves pública y privada.

#Creating a new rsa key pair with empty password  
!ssh-keygen -t rsa -P "" -f ~/.ssh/id\_rsa

#Showing the public key  
!more /root/.ssh/id\_rsa.pub

#Copying the key to autorized keys  
!cat HOME/.ssh/idrse.pub>>HOME/.ssh/authorized\_keys  
#Changing the permissions on the key  
!chmod 0600 ~/.ssh/authorized\_keys

#Conneting with the local machine  
!ssh -o StrictHostKeyChecking=no localhost uptime

**Instalar Hadoop 3.2.3**

Descarga y configura el entorno de Hadoop.

#Downloading Hadoop 3.2.3  
!wget -q https://archive.apache.org/dist/hadoop/common/hadoop-3.2.3/hadoop-3.2.3.tar.gz

#Untarring the file  
!sudo tar -xzf hadoop-3.2.3.tar.gz

#Removing the tar file  
!rm hadoop-3.2.3.tar.gz

La ubicación estándar para instalar Hadoop es

/usr/local  
/opt

#Copying the hadoop files to user/local  
!cp -r hadoop-3.2.3/ /usr/local/  
#-r copy directories recursively

#Exploring hadoop-3.2.3/etc/hadoop directory  
!ls /usr/local/hadoop-3.2.3/etc/hadoop  
#we can see various configuration files of hadoop

Se requiere configurar algunas cosas antes de ejecutar Hadoop. Es decir, se requiere agregar o modificar algunos parámetros en estos archivos de configuración para operar Hadoop en el modo que queramos.

Configuración del archivo hadoop-env.sh

hadoop-env.sh es un script bash que contiene variables de entorno que se usan en los scripts para ejecutar Hadoop

#Exploring hadoop-env.sh file  
!cat /usr/local/hadoop-3.2.3/etc/hadoop/hadoop-env.sh

La única variable de entorno requerida es JAVA\_HOME. Todas las demás son opcionales.

Para especificar la variable JAVA\_HOME en hadoop-env.sh, debemos descomentar la línea de exportación y actualizarla con el directorio actual.

En este caso, debería verse así:

export JAVA\_HOME=/usr/lib/jvm/java-8-openjdk-amd64

#Adding JAVA\_HOME directory to hadoop-env.sh file  
!sed -i '/export JAVA\_HOME=/a export JAVA\_HOME=\/usr\/lib\/jvm\/java-8-openjdk-amd64' /usr/local/hadoop-3.2.3/etc/hadoop/hadoop-env.sh

Porque es conveniente se crea una variable de entorno que apunta al directorio de instalación de Hadoop

#Creating Hadoop home variable  
os.environ["HADOOP\_HOME"] = "/usr/local/hadoop-3.2.3"

Configuración de archivos XML

La mayoría de las configuraciones de Hadoop se encuentran en archivos de configuración XML. Estos archivos también se conocen como recursos.

Tienen la siguiente estructura:

<configuration>  
...

<property>  
<name>...</name>  
<value>...</value>  
<description>...</description>  
</property>

...  
</configuration>

El archivo XLM puede contener cualquier cantidad de elementos de propiedad. Cada elemento de propiedad define un par de nombre-valor de configuración específico.

La configuración de Hadoop se controla mediante dos tipos distintos de archivos de configuración XLM:

* Predeterminados (solo lectura): core-default.xml, hdfs-default.xml, mapred-default.xml, yarn-default.xml. Estos archivos nunca deben modificarse.
* Archivos de configuración específicos del sitio: core-site.xml, hdfs-site.xml, mapred-site.xml, yarn-site.xml. Estos archivos se cargan desde la ruta de clase y sus valores se utilizan para sobrescribir los valores correspondientes de las propiedades en los archivos de configuración predeterminados coincidentes.

#Exploring hadoop-3.2.3/etc/hadoop xml files  
!ls $HADOOP\_HOME/etc/hadoop/\*.xml

Cada componente de Hadoop se configura mediante un archivo xml

* core-site.xml: propiedades comunes
* hdfs-site.xml: propiedades de HDFS
* mapred-site.xml: propiedades de MapReduce
* yarn-site.xml: propiedades de YARN

Al configurar estos archivos xml como corresponde, Hadoop se puede ejecutar en uno de los tres modos.

#Content of core-site.xml file  
!cat $HADOOP\_HOME/etc/hadoop/core-site.xml

Si las propiedades están vacías de forma predeterminada, no hay nada que sobrescribir y Hadoop se ejecuta con las propiedades predeterminadas.

**Ejecución de Hadoop en modo independiente**

Con las propiedades de configuración predeterminadas, Hadoop se ejecuta en modo independiente (modo no distribuido). Es decir, el modo independiente (también conocido como modo local) es el modo predeterminado para Hadoop.

No hay deamons para ejecutar. Solo se utiliza un único proceso de Java.

Se utilizan el sistema de archivos local y el ejecutor de trabajos de MapReduce local.

El comando para ejecutar un programa mapreduce de Hadoop escrito en Java es:

$HADOOP\_HOME/bin/hadoop jar <jar>

jar es una herramienta de archivo de Java que empaqueta (y comprime) un conjunto de archivos en un solo archivo.

La instalación predeterminada ya tiene varios programas de ejemplo de MapReduce que se pueden usar.

#Exploring mapreduce tools  
!ls $HADOOP\_HOME/share/hadoop/mapreduce/\*.jar

**Ejecución de Hadoop en modo pseudodistribuido**

Para simular un entorno distribuido usando recursos locales.

En el modo pseudodistribuido entran en juego todos los componentes distribuidos de Hadoop. Es decir, todos los demons de Hadoop que son responsables del almacenamiento distribuido y del procesamiento distribuido se ejecutarán en la misma máquina.

Master demons:

* NameNode
* Resource Manager
* NameNode Standby

Slave deamons:

* DataNode
* Node Manager

Configuración de archivos XML

Como se mencionó, al configurar las propiedades en los archivos de configuración xml del sitio, se sobrrescriben las propiedades correspondientes en los archivos de configuración xml predeterminados y, de esta manera, se le indica a Hadoop qué máquinas están en el clúster y dónde y cómo se quiere ejecutar los demons de Hadoop.

El contenido específico que deben tener estos archivos para que Hadoop se ejecute en modo pseudodistribuido se puede encontrar en la documentación de la versión en el sitio web oficial. Para Hadoop 3.2.3, el sitio web es: <https://hadoop.apache.org/docs/r3.2.3/hadoop-project-dist/hadoop-common/SingleCluster.html>

Configuración de core-site.xml

#Adding required property to core-site.xlm file  
!sed -i '//a\  
 \n\  
 fs.defaultFS\n\  
 hdfs://localhost:9000\n\  
 ' \  
$HADOOP\_HOME/etc/hadoop/core-site.xml

#Content of core-site.xml after the editing  
!cat $HADOOP\_HOME/etc/hadoop/core-site.xml

Configuración de hdfs-site.xml

#Adding required property to hdfs-site.xml file  
#Since we are running Hadoop in only one machine, a replication factor greater than 1 does not make sense   
!sed -i '//a\  
 \n\  
 dfs.replication\n\  
 1\n\  
 ' \  
$HADOOP\_HOME/etc/hadoop/hdfs-site.xml

#Content of hdfs-site.xml after the editing  
!cat $HADOOP\_HOME/etc/hadoop/hdfs-site.xml

Configuración de mapred-site.xml

#Adding required properties to mapred-site.xml file  
!sed -i '//a\  
 \n\  
 mapreduce.framework.name\n\  
 yarn\n\  
 \n\  
 \n\  
 mapreduce.application.classpath\n\  
HADOOPMAPREDHOME/share/hadoop/mapreduce/\*:HADOOP\_MAPRED\_HOME/share/hadoop/mapreduce/lib/\*\n\  
 ' \  
$HADOOP\_HOME/etc/hadoop/mapred-site.xml

#Content of mapred-site.xml after the editing  
!cat $HADOOP\_HOME/etc/hadoop/mapred-site.xml

Configuración de yarn-site.xml

#Adding required properties to yarn-site.xml file  
!sed -i '//a\  
 \n\  
 The hostname of the RM.\n\  
 yarn.resourcemanager.hostname\n\  
 localhost\n\  
 \n\  
 \n\  
 yarn.nodemanager.aux-services\n\  
 mapreduce\_shuffle\n\  
 \n\  
 \n\  
 yarn.nodemanager.env-whitelist\n\  
 JAVA\_HOME,HADOOP\_COMMON\_HOME,HADOOP\_HDFS\_HOME,HADOOP\_CONF\_DIR,CLASSPATH\_PREPEND\_DISTCACHE,HADOOP\_YARN\_HOME,HADOOP\_HOME,PATH,LANG,TZ,HADOOP\_MAPRED\_HOME\n\  
 ' \  
$HADOOP\_HOME/etc/hadoop/yarn-site.xml

#Content of yarn-site.xml after the editing  
!cat $HADOOP\_HOME/etc/hadoop/yarn-site.xml

Formateo del sistema de archivos HDFS

Antes de poder utilizar HDFS por primera vez, se debe formatear el sistema de archivos. El proceso de formateo crea un sistema de archivos vacío mediante la creación de los directorios de almacenamiento y las versiones iniciales de los NameNodes.

!$HADOOP\_HOME/bin/hdfs namenode -format

Scripts de Hadoop

Hadoop incluye scripts para ejecutar comandos e iniciar y detener daemons en todo el clúster. Estos scripts se pueden encontrar en los directorios bin y sbin

#Exploring Hadoop scripts available in sbin directory  
!ls $HADOOP\_HOME/sbin

#Creating other necessary enviroment variables before starting nodes  
os.environ["HDFS\_NAMENODE\_USER"] = "root"  
os.environ["HDFS\_DATANODE\_USER"] = "root"  
os.environ["HDFS\_SECONDARYNAMENODE\_USER"] = "root"  
os.environ["YARN\_RESOURCEMANAGER\_USER"] = "root"  
os.environ["YARN\_NODEMANAGER\_USER"] = "root"

#Launching hdfs deamons  
!$HADOOP\_HOME/sbin/start-dfs.sh

#Listing the running deamons  
!jps

#Launching yarn deamons  
#nohup causes a process to ignore a SIGHUP signal  
!nohup $HADOOP\_HOME/sbin/start-yarn.sh

#Listing the running deamons  
!jps

Supervisión del clúster Hadoop con comandos de administración de Hadoop

#Report the basic file system information and statistics  
!$HADOOP\_HOME/bin/hdfs dfsadmin -report

Monitoreo del clúster Hadoop con la interfaz del navegador

from google.colab import output

#The namenode posts the general report on port 9870  
output.serve\_kernel\_port\_as\_window(9870)

Ejecución del recuento de palabras en modo pseudodistribuido

#Dowloading text example to use as input (if it has not been donwloaded yet)  
!wget -q https://www.mirrorservice.org/sites/ftp.ibiblio.org/pub/docs/books/gutenberg/1/0/101/101.txt

#Creating directory in HDFS  
!$HADOOP\_HOME/bin/hdfs dfs -mkdir /word\_count  
#Coping file from local file system to HDFS  
!$HADOOP\_HOME/bin/hdfs dfs -put /content/101.txt /word\_count

#Exploring Hadoop folder  
!$HADOOP\_HOME/bin/hdfs dfs -ls /word\_count

#Running MapReduce program wordcount  
!$HADOOP\_HOME/bin/hadoop jar $HADOOP\_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.2.3.jar wordcount /word\_count/101.txt /word\_count/output/

#Exploring the created output directory  
#part-r-00000 contains the actual ouput  
!$HADOOP\_HOME/bin/hdfs dfs -ls /word\_count/output

#Printing out first 50 lines  
!$HADOOP\_HOME/bin/hdfs dfs -cat /word\_count/output/part-r-00000 | head -50

Transmisión de Hadoop con Python

La transmisión de Hadoop es una función que viene con Hadoop y permite a los usuarios o desarrolladores utilizar distintos lenguajes para escribir programas MapReduce, como Python, C++, Ruby, etc.

La utilidad creará un trabajo Map/Reduce, lo enviará a un clúster adecuado y supervisará el progreso del trabajo hasta que se complete.

#Exploring Hadoop utilities available  
!ls $HADOOP\_HOME/share/hadoop/tools/lib/

#Dowloading text example to use as input (if it has not been donwloaded yet)  
!wget -q https://www.mirrorservice.org/sites/ftp.ibiblio.org/pub/docs/books/gutenberg/1/0/101/101.txt

#Creating directory in HDFS  
!$HADOOP\_HOME/bin/hdfs dfs -mkdir /word\_count\_with\_python

#Copying the file from local file system to Hadoop distributed file system (HDFS)  
!$HADOOP\_HOME/bin/hdfs dfs -put /content/101.txt /word\_count\_with\_python

Mapper

El mapper es un ejecutable que lee todos los registros de entrada de uno o más archivos y genera una salida en forma de pares clave-valor que funciona como entrada para el reductor.

%%writefile mapper.py

#!/usr/bin/env python

#'#!' is known as shebang and used for interpreting the script

# import sys because we need to read and write data to STDIN and STDOUT

import sys  
# reading entire line from STDIN (standard input)  
for line in sys.stdin:  
 # to remove leading and trailing whitespace  
 line = line.strip()  
 # split the line into words  
 words = line.split()

# we are looping over the words array and printing the Word  
 # with the count of 1 to the STDOUT  
 for word in words:  
 # write the results to STDOUT (standard output);  
 # what we output here will be the input for the  
 # Reduce step, i.e. the input for reducer.py  
 print('%s\t%s' % (word, 1))

Reducer

El reducer es un ejecutable que lee todos los pares clave-valor intermedios generados por el asignador y genera una salida final como resultado de una operación de cálculo como la adición, el filtrado y la agregación.

Tanto el asignador como el reducer leen la entrada de la entrada estándar (línea por línea) y emiten la salida a la salida estándar.

%%writefile reducer.py

#!/usr/bin/env python

from operator import itemgetter

import sys  
current\_word = None  
current\_count = 0  
word = None

# read the entire line from STDIN

for line in sys.stdin:  
 # remove leading and trailing whitespace  
 line = line.strip()  
 # splitting the data on the basis of tab we have provided in mapper.py

word, count = line.split('\t', 1)  
 # convert count (currently a string) to int  
 try:  
 count = int(count)  
 except ValueError:  
 # count was not a number, so silently  
 # ignore/discard this line  
 continue

# this IF-switch only works because Hadoop sorts map output  
 # by key (here: word) before it is passed to the reducer  
 if current\_word == word:  
 current\_count += count  
 else:  
 if current\_word: #to not print current\_word=None  
 # write result to STDOUT  
 print('%s\t%s' % (current\_word, current\_count))  
 current\_count = count  
 current\_word = word

# do not forget to output the last word if needed!  
if current\_word == word:  
 print('%s\t%s' % (current\_word, current\_count))

#Testing our MapReduce job locally (Hadoop does not participate here)  
!cat 101.txt | python mapper.py | sort -k1,1 | python reducer.py | head -50  
#We apply sorting after the mapper because it is the default operation in MapReduce architecture

#Changing the permissions of the files  
!chmod 777 /content/mapper.py /content/reducer.py  
#Setting 777 permissions to a file or directory means that it will be readable, writable and executable by all users

#Running MapReduce programs  
!$HADOOP\_HOME/bin/hadoop jar $HADOOP\_HOME/share/hadoop/tools/lib/hadoop-streaming-3.2.3.jar \  
 -input /word\_count\_with\_python/101.txt  
 -output /word\_count\_with\_python/output \  
 -mapper "python /content/mapper.py" \  
 -reducer "python /content/reducer.py"

**Consigna 3**

Actualizar el sistema

sudo apt -get update

Y tener Python instalado

sudo apt -get install python  
python --versión

Buscar o instalar la máquina virtual de java

sudo apt -get install default-jre  
java -versión

Dirigirse a la página de Spark: <https://saprk.apache.org/downloads.html> , elegir la versión 3.1.2 el paquete y descargar el comprimido .tgz

Ejecutar por consola, en la ruta donde se encuentra el archivo descargado. Y moverlo al directorio opt y navegar hasta allí

ll |grep spark  
sudo mv spark-3.1.2-bin-hadoop3.2.tgz /opt/  
cd /opt/  
ll

Extraer el archivo del comprimido, navegar a la carpeta, y mostrar la ruta en la que se encuentra la consola.

sudo tar –zxvf spark-3.1.2-bin-hadoop3.2.tgz  
cd spark-3.1.2-bin-hadoop3.2/  
pwd

Copiar la ruta mostrada. Ahora editar el archivo profile.

sudo vim /etc/profile

Pegar la ruta copiada. Las últimas líneas deben quedar así:

export SPARK\_HOME=/opt/spark-3.1.2-bin-hadoop3.2  
export PATH=$PATH:$SPARK\_HOME/bin:$SPARK\_HOME/sbin

Con :wq! para guardar y salir. Reiniciar o cerrar la sesión y volver a iniciarla. Luego de eso spark ya debería estar instalado. Verificar la versión instalada con:

pyspark