

Examen Final

Nombre: LEONEL ALVARO CHAMACA LIMA

C.I: 6853618 LP

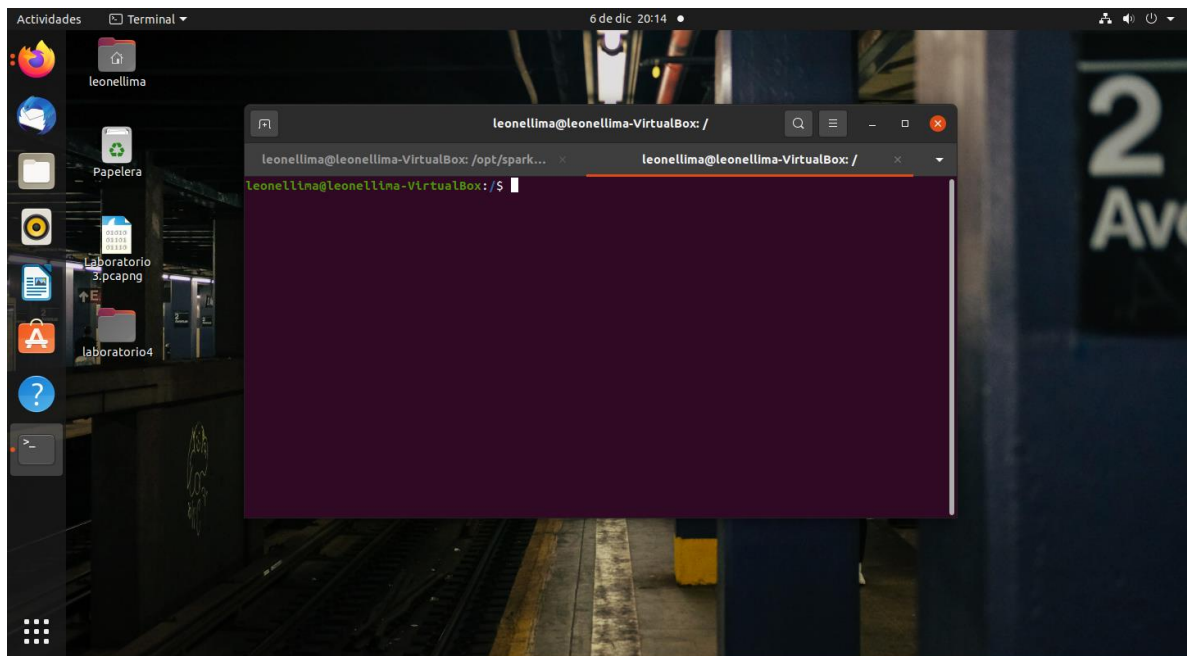
1. En una máquina virtual realice la configuración de apache spark, puede guiarse en cualquier tutorial o el proporcionado por el docente.

url: <https://computingforgeeks.com/how-to-install-apache-spark-on-ubuntu-debian/>

Con el shell podra ejecutar scala por defecto

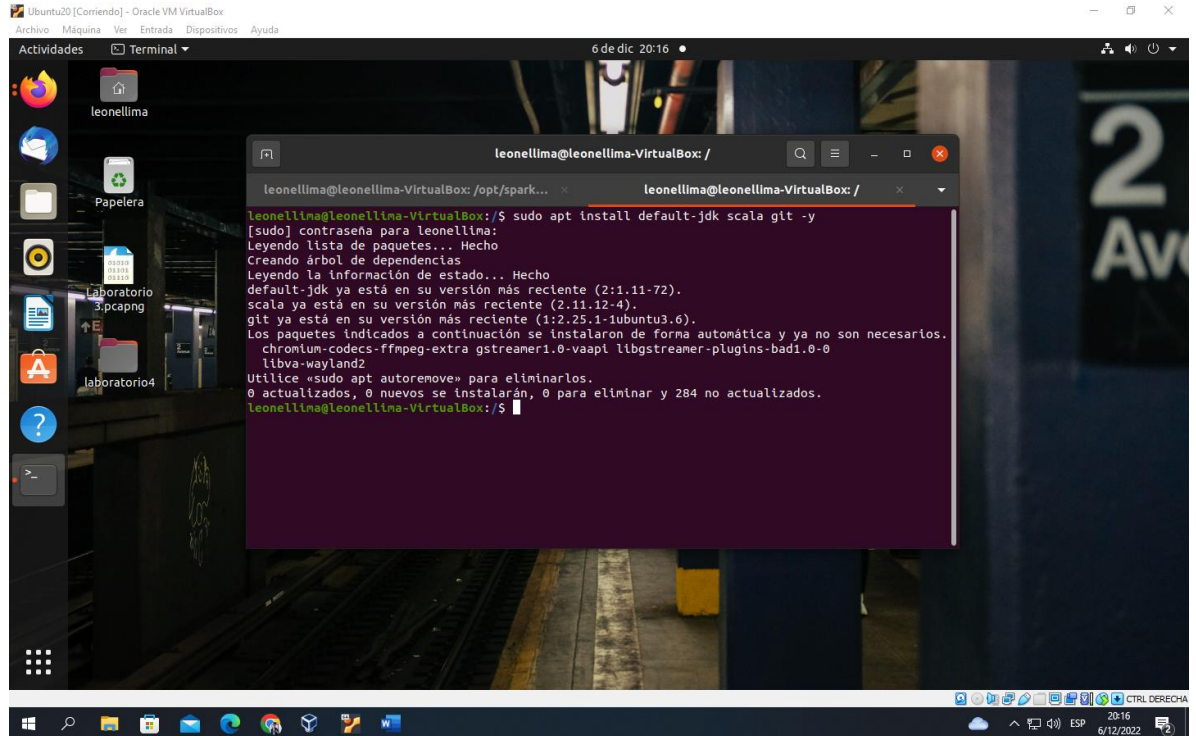
Instale Python para spark

1. INICIAR LA MAQUINA VIRTUAL
INSTALADO EL UBUNTU 20



2. ENTRAR A LA CONSOLA EN INSERTAR LA SIGUIENTE LINEA DE CODIGO

`sudo apt install default-jdk scala git -y`

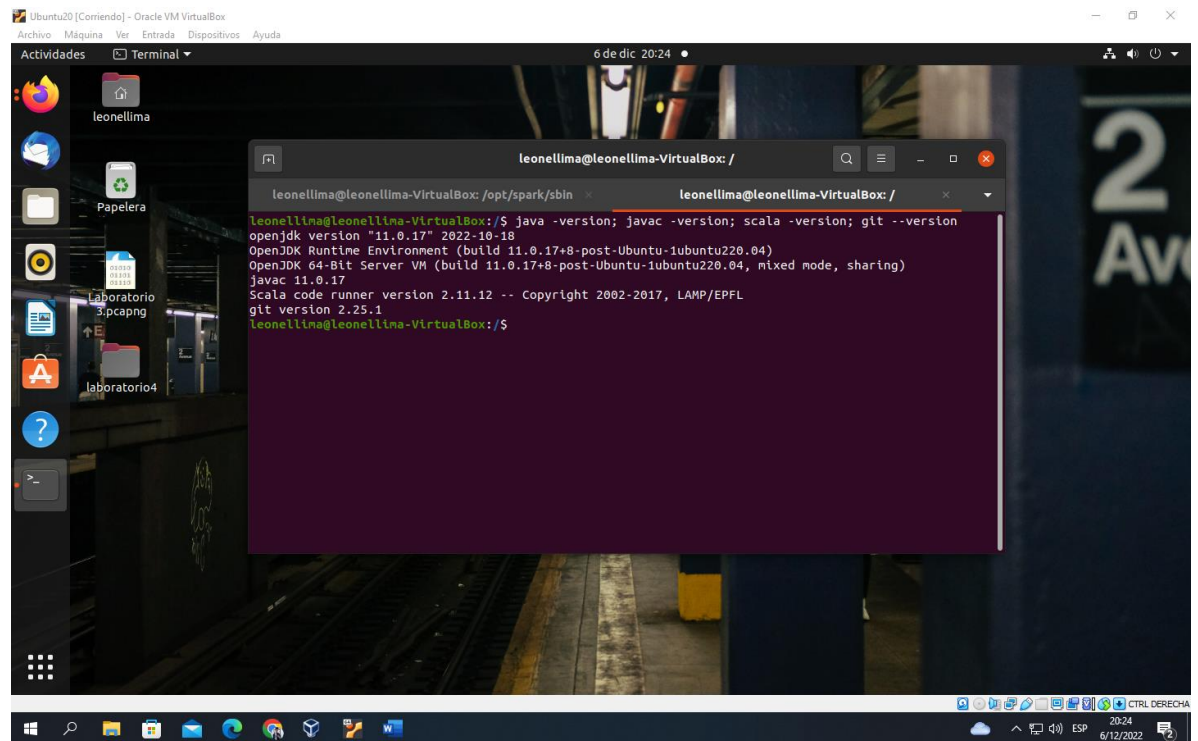


```
leonellima@leonellima-VirtualBox: /opt/spark... leonellima@leonellima-VirtualBox: /
leonellima@leonellima-VirtualBox:/$ sudo apt install default-jdk scala git -y
[sudo] contraseña para leonellima:
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
default-jdk ya está en su versión más reciente (2:1.11-72).
scala ya está en su versión más reciente (2.11.12-4).
git ya está en su versión más reciente (1:2.25.1-1ubuntu3.6).
Los paquetes indicados a continuación se instalaron de forma automática y ya no son necesarios.
chromium-codecs-ffmpeg-extra gstreamer1.0-vaapi libgstreamer-plugins-bad1.0-0
libva-wayland2
Utilice «sudo apt autoremove» para eliminarlos.
0 actualizados, 0 nuevos se instalarán, 0 para eliminar y 284 no actualizados.
leonellima@leonellima-VirtualBox:/$
```

Antes de descargar y configurar Spark, necesitamos instalar dependencias. Estos pasos incluyen la instalación de la siguiente paquetería. JDK, Scala Git, Ya que lo tengo instalado, me sale ese mensaje, al contrario si no lo tendría instalado, empezaría la descarga.

3. Verificamos la versión que tenemos con el comando

```
java -version; javac -version; scala -version; git -version
```



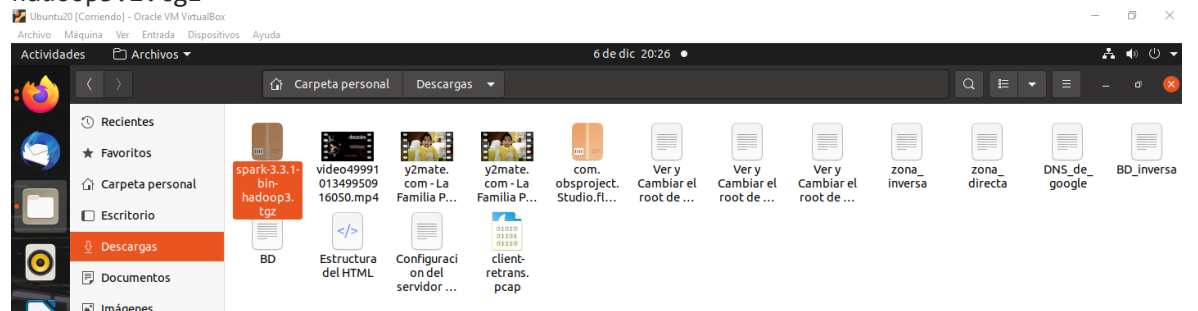
The screenshot shows a terminal window titled 'leonellima@leonellima-VirtualBox: /' with the command 'java -version; javac -version; scala -version; git --version' executed. The output is as follows:

```
leonellima@leonellima-VirtualBox:/$ java -version; javac -version; scala -version; git --version
openjdk version "11.0.17" 2022-10-18
OpenJDK Runtime Environment (build 11.0.17+8-post-Ubuntu-1ubuntu220.04)
OpenJDK 64-Bit Server VM (build 11.0.17+8-post-Ubuntu-1ubuntu220.04, mixed mode, sharing)
javac 11.0.17
Scala code runner version 2.11.12 -- Copyright 2002-2017, LAMP/EPFL
git version 2.25.1
leonellima@leonellima-VirtualBox:/$
```

Verificamos la versión para continuar.

4. DESCARGAMOS SPARK POR EL SIGUIENTE COMANDO

```
wget https://d1cdn.apache.org/spark/spark-3.2.1/spark-3.2.1-bin-hadoop3.2.tgz
```



Lo que hará será descargar la carpeta spark y la versión deseada

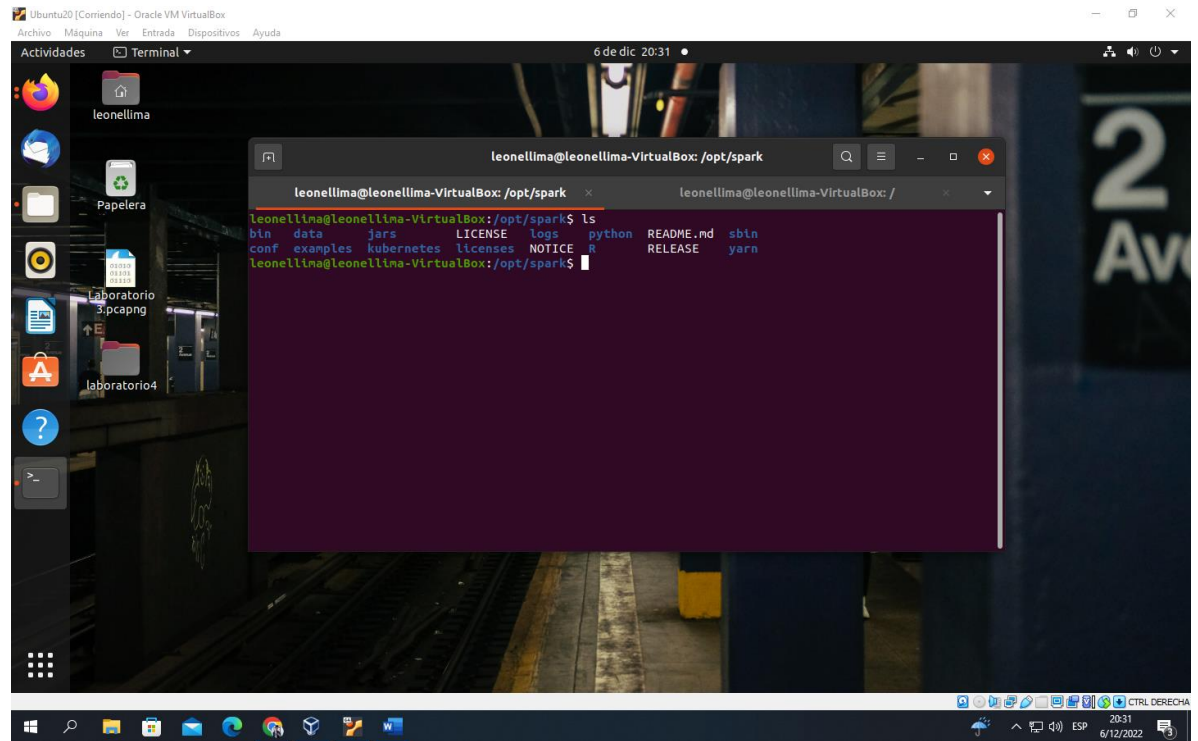
5. Descomprimos con el comando

```
tar xvf spark-*
```

Recordemos que una vez descomprimido tenemos que buscar la carpeta SPARK para trasladarla al OTC

Con el siguiente comando

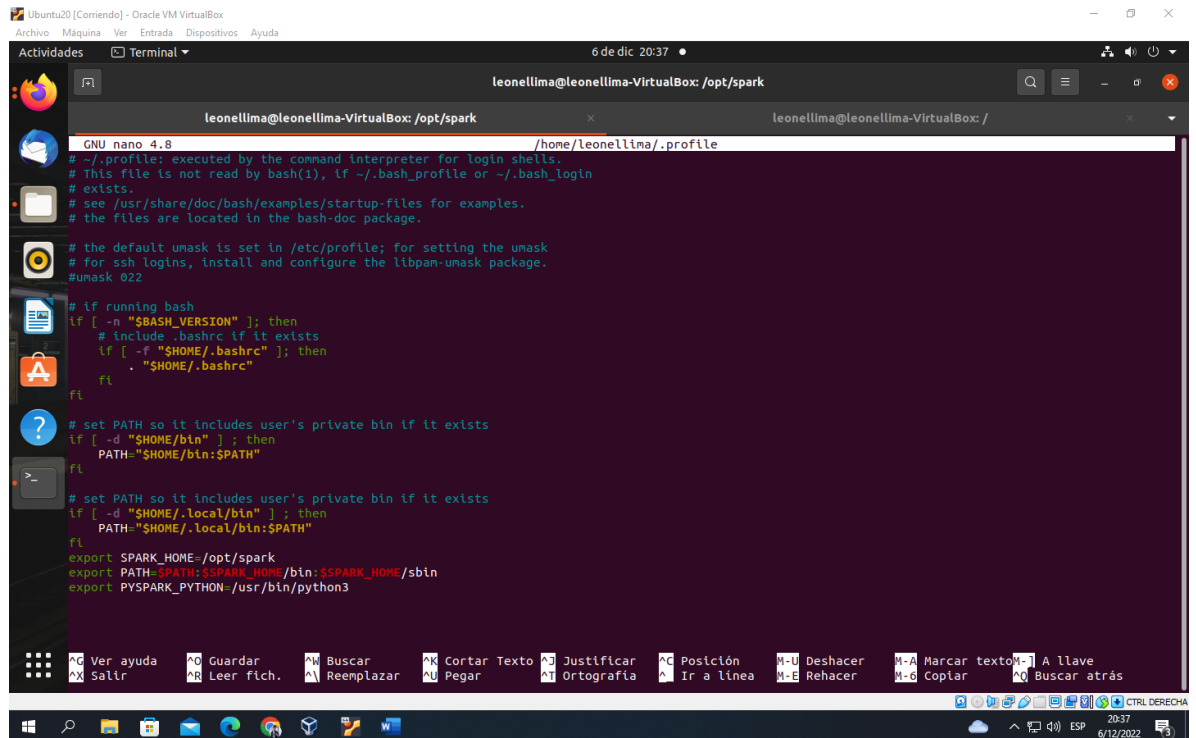
```
sudo mv spark-3.2.1-bin-hadoop3.2 /opt/spark
```



Ya dentro podremos observar todas las carpetas dentro del spark

6. CREAMOS UN ENTORNO PARA SPARK

```
nano ~/.profile
```



The screenshot shows a terminal window titled 'leonellima@leonellima-VirtualBox: /opt/spark'. The terminal is running the nano 4.8 text editor, editing the file '/home/leonellima/.profile'. The file contains the following content:

```
# ~/.profile: executed by the command interpreter for login shells.
# This file is not read by bash(1), if ~/.bash_profile or ~/.bash_login
# exists.
# see /usr/share/doc/bash/examples/startup-files for examples.
# the files are located in the bash-doc package.

# the default umask is set in /etc/profile; for setting the umask
# for ssh logins, install and configure the libpam-umask package.
umask 022

# if running bash
if [ -n "$BASH_VERSION" ]; then
    # include .bashrc if it exists
    if [ -f "$HOME/.bashrc" ]; then
        . "$HOME/.bashrc"
    fi
fi

# set PATH so it includes user's private bin if it exists
if [ -d "$HOME/bin" ]; then
    PATH="$HOME/bin:$PATH"
fi

# set PATH so it includes user's private bin if it exists
if [ -d "$HOME/.local/bin" ]; then
    PATH="$HOME/.local/bin:$PATH"
fi

export SPARK_HOME=/opt/spark
export PATH=$PATH:$SPARK_HOME/bin:$SPARK_HOME/sbin
export PYSPARK_PYTHON=/usr/bin/python3
```

The terminal window also shows a menu bar at the bottom with various keyboard shortcuts for editing and navigation.

Copiamos las siguientes líneas de código

```
echo "export SPARK_HOME=/opt/spark" >> ~/.profile
```

```
echo "export PATH=$PATH:$SPARK_HOME/bin:$SPARK_HOME/sbin" >> ~/.profile
```

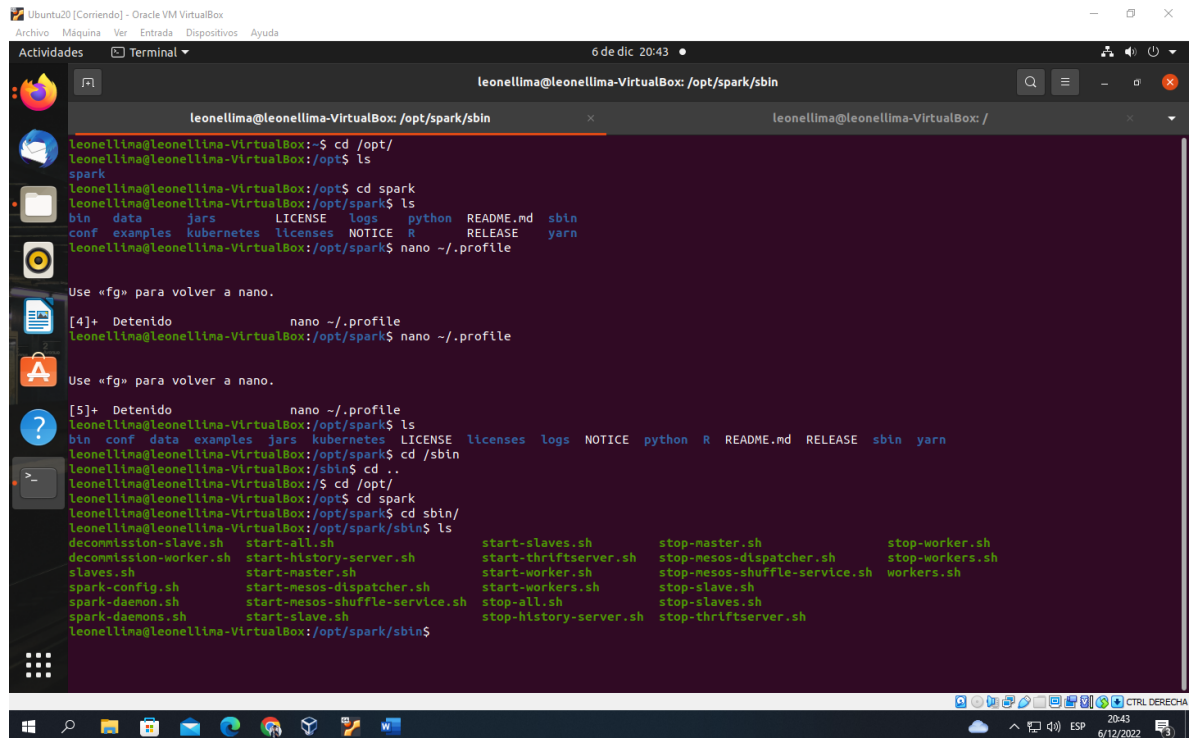
```
echo "export PYSPARK_PYTHON=/usr/bin/python3" >> ~/.profile
```

Despues introducimos la siguiente código

```
source ~/.profile
```

para que se guarde.

7. Ahora iniciamos la vertificacion de la instalación de SPARK

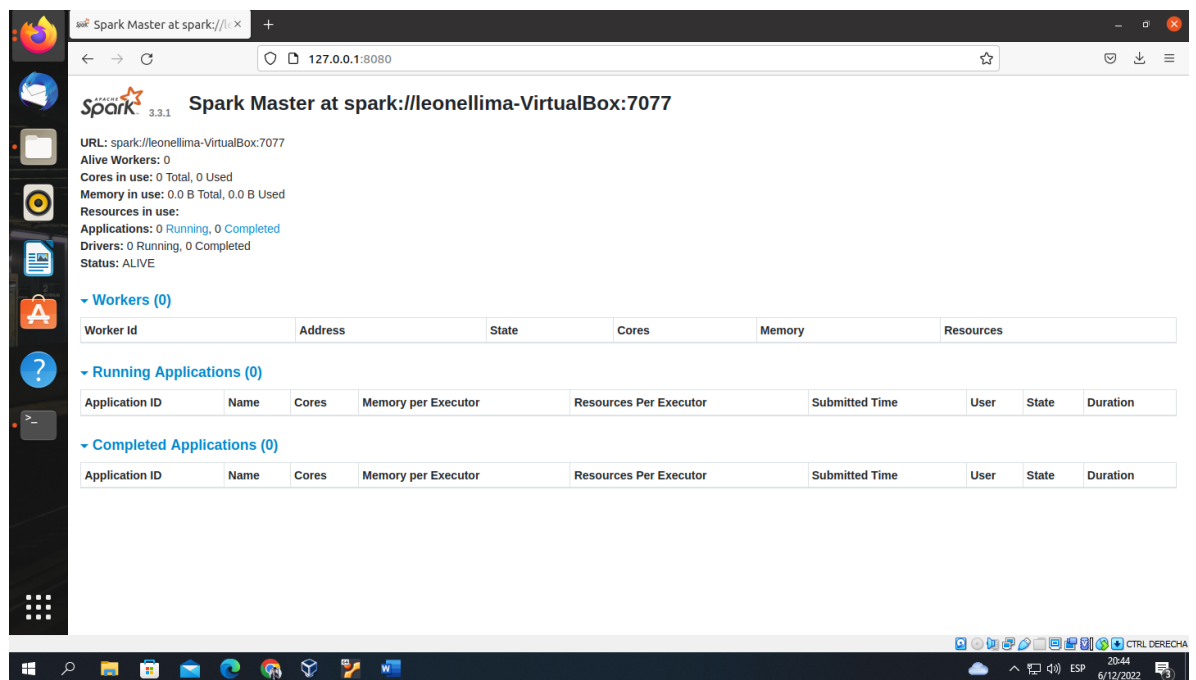


```
leonellima@leonellima-VirtualBox: /opt/spark/sbin
leonellima@leonellima-VirtualBox:~$ cd /opt/
leonellima@leonellima-VirtualBox:/opt$ ls
spark
leonellima@leonellima-VirtualBox:/opt$ cd spark
leonellima@leonellima-VirtualBox:/opt/spark$ ls
bin  data  jars  LICENSE  logs  python  README.md  sbin
conf  examples  kubernetes  licenses  NOTICE  R  RELEASE  yarn
leonellima@leonellima-VirtualBox:/opt/spark$ nano ~/.profile
Use «fg» para volver a nano.
[4]+ Detenido      nano ~/.profile
leonellima@leonellima-VirtualBox:/opt/spark$ nano ~/.profile
Use «fg» para volver a nano.
[5]+ Detenido      nano ~/.profile
leonellima@leonellima-VirtualBox:/opt/spark$ ls
bin  conf  data  examples  jars  kubernetes  LICENSE  licenses  logs  NOTICE  python  R  README.md  RELEASE  sbin  yarn
leonellima@leonellima-VirtualBox:/opt/spark$ cd /sbin
leonellima@leonellima-VirtualBox:/sbin$ cd ..
leonellima@leonellima-VirtualBox:$ cd /opt/
leonellima@leonellima-VirtualBox:/opt$ cd spark
leonellima@leonellima-VirtualBox:/opt/spark$ cd sbin/
leonellima@leonellima-VirtualBox:/opt/spark/sbin$ ls
decommission-slave.sh      start-all.sh              start-slaves.sh            stop-master.sh             stop-worker.sh
decommission-worker.sh    start-history-server.sh    start-thriftserver.sh      stop-mesos-dispatcher.sh   stop-workers.sh
slaves.sh                 start-master.sh            start-workers.sh           stop-mesos-shuffle-service.sh  workers.sh
spark-config.sh           start-mesos-dispatcher.sh  start-workers.sh           stop-slave.sh             
spark-daemon.sh           start-mesos-shuffle-service.sh  stop-all.sh              stop-slaves.sh             
spark-daemons.sh         start-slave.sh             stop-history-server.sh     stop-thriftserver.sh
```

Para verificar si ya esta instalado correctamente activaremos el script que se encuentra en la carpeta SBIN, ingresamos e activamos el script

“start-master.sh”

8. AHORA ENTRAMOS AL NAVEGADOR LOCALHOST



Spark Master at spark://leonellima-VirtualBox:7077

URL: spark://leonellima-VirtualBox:7077

Alive Workers: 0

Cores in use: 0 Total, 0 Used

Memory in use: 0.0 B Total, 0.0 B Used

Resources in use:

Applications: 0 Running, 0 Completed

Drivers: 0 Running, 0 Completed

Status: ALIVE

Workers (0)

Worker Id	Address	State	Cores	Memory	Resources
-----------	---------	-------	-------	--------	-----------

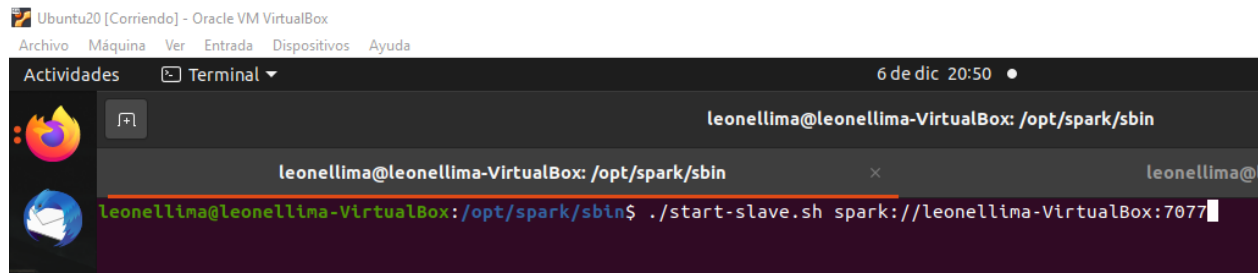
Running Applications (0)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
----------------	------	-------	---------------------	------------------------	----------------	------	-------	----------

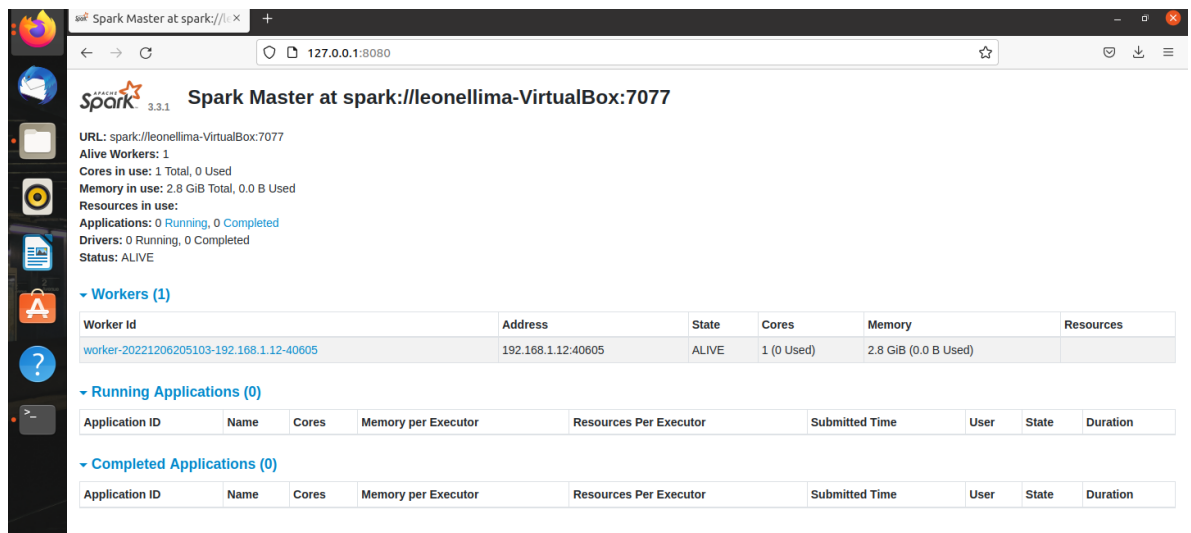
Completed Applications (0)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
----------------	------	-------	---------------------	------------------------	----------------	------	-------	----------

9. ACTIVAMOS EL ESCLAVO



COPIAMOS LA URL QUE NOS MUESTRA LA VENTANA DEL SPARK



Ya tenemos el esclavo funcionando

Ahora ya vemos que funciona iniciamos con la SPARK SHELL

10. Entramos a BIN para activar el script Shell


```
leonellima@leonellima-VirtualBox: /opt/spark/bin$ ls
beeline find-spark-home.cmd pyspark2.cmd spark-class sparkR2.cmd spark-shell.cmd spark-submit
beeline.cmd load-spark-env.cmd pyspark.cmd spark-class2.cmd sparkR.cmd spark-sql spark-submit2.cmd
docker-image-tool.sh load-spark-env.sh run-example spark-class.cmd spark-shell spark-sql2.cmd spark-submit.cmd
find-spark-home pyspark run-example.cmd sparkR spark-shell2.cmd spark-sql.cmd

leonellima@leonellima-VirtualBox: /opt/spark/bin$ ./spark-shell
22/12/06 21:00:11 WARN Utils: Your hostname, leonellima-VirtualBox resolves to a loopback address: 127.0.1.1; using 192.168.1.12 instead (on interface
enp0s3)
22/12/06 21:00:11 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
22/12/06 21:00:25 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Spark context Web UI available at http://192.168.1.12:4040
Spark context available as 'sc' (master = local[*], app id = local-1670374827381).
Spark session available as 'spark'.
Welcome to

      _/  __|  _/  _/
     /_  /  _/  _/  _/  version 3.3.1

Using Scala version 2.12.15 (OpenJDK 64-Bit Server VM, Java 11.0.17)
Type in expressions to have them evaluated.
Type :help for more information.

scala>
```

Y ahora con pyspark

```
scala> !q
leonellima@leonellima-VirtualBox: /opt/spark/bin$ ls
beeline find-spark-home.cmd pyspark2.cmd spark-class sparkR2.cmd spark-shell.cmd spark-submit
beeline.cmd load-spark-env.cmd pyspark.cmd spark-class2.cmd sparkR.cmd spark-sql spark-submit2.cmd
docker-image-tool.sh load-spark-env.sh run-example spark-class.cmd spark-shell spark-sql2.cmd spark-submit.cmd
find-spark-home pyspark run-example.cmd sparkR spark-shell2.cmd spark-sql.cmd

leonellima@leonellima-VirtualBox: /opt/spark/bin$ pyspark
Python 3.8.10 (default, Jun 22 2022, 20:18:18)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license()" for more information.
22/12/06 21:07:06 WARN Utils: Your hostname, leonellima-VirtualBox resolves to a loopback address: 127.0.1.1; using 192.168.1.12 instead (on interface
enp0s3)
22/12/06 21:07:06 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
22/12/06 21:07:08 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Welcome to

      _/  __|  _/  _/
     /_  /  _/  _/  _/  version 3.3.1

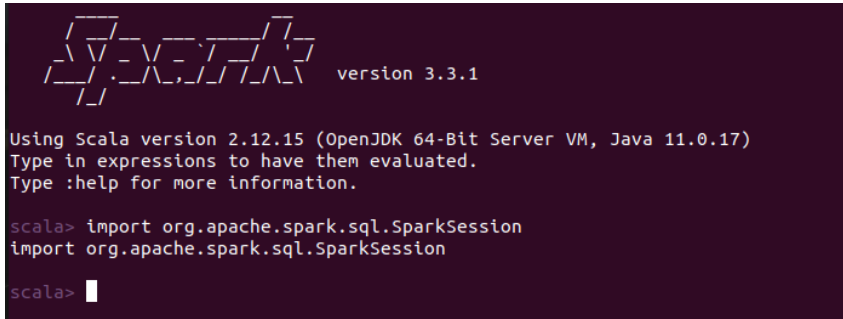
Using Python version 3.8.10 (default, Jun 22 2022 20:18:18)
Spark context Web UI available at http://192.168.1.12:4040
Spark context available as 'sc' (master = local[*], app id = local-1670375230614).
SparkSession available as 'spark'.
>>>
```


2. Realice el siguiente código, documente su funcionamiento en apache spark

Sesiones

Paso 1: Debemos importar las siguientes librerias

```
import org.apache.spark.sql.SparkSession
```



```
version 3.3.1


Using Scala version 2.12.15 (OpenJDK 64-Bit Server VM, Java 11.0.17)
Type in expressions to have them evaluated.
Type :help for more information.

scala> import org.apache.spark.sql.SparkSession
import org.apache.spark.sql.SparkSession

scala> 
```

Paso 2: Iniciaremos colocando la primera linea de codigo

```
val spark: SparkSession = SparkSession.builder()
    .master("local[*]")
    .appName("simple-app")
    .getOrCreate()
```



```
scala> val spark = SparkSession
spark: org.apache.spark.sql.SparkSession.type = org.apache.spark.sql.SparkSession$@77d50cde

scala> .builder()
res0: spark.Builder = org.apache.spark.sql.SparkSession$Builder@399ebdee

scala> .appName("Spark SQL basic example")
res1: spark.Builder = org.apache.spark.sql.SparkSession$Builder@399ebdee

scala> .config("spark.some.config.option", "some-value")
res2: spark.Builder = org.apache.spark.sql.SparkSession$Builder@399ebdee

scala> .getOrCreate()
22/12/11 18:46:18 WARN SparkSession: Using an existing Spark session; only runtime SQL configurations will take effect.
res3: org.apache.spark.sql.SparkSession = org.apache.spark.sql.SparkSession@17a394d2
```

El punto de entrada a la funcionalidad de la funcionalidad de Spark el comando SparkSession.builder es para crear una session basica.

Paso 3: colocamos la siguiente linea de codigo

```
val dataSet: Dataset[String] = spark.read.textFile("textfile.csv")
val df: DataFrame = dataSet.toDF()
```

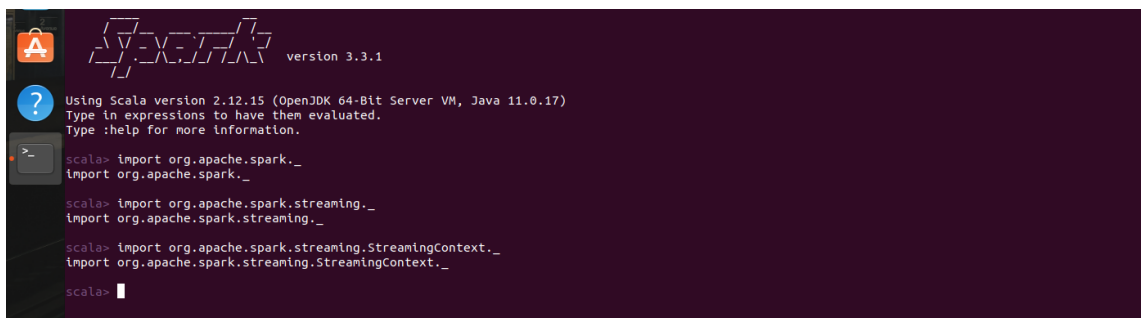
Para poder leer un archivo con extencion “.csv” y lo mostramos.

```
val spark: SparkSession = SparkSession.builder()  
  .master("local[*]")  
  .appName("simple-app")  
  .getOrCreate()  
  
val dataSet: Dataset[String] = spark.read.textFile("textfile.csv")  
val df: DataFrame = dataSet.toDF()
```

Streaming

Paso 1: Debemos importar las siguientes librerias

```
import org.apache.spark._  
import org.apache.spark.streaming._  
import org.apache.spark.streaming.StreamingContext._
```



The screenshot shows a Scala REPL window with the following content:

```
scala> import org.apache.spark._  
import org.apache.spark._  
  
scala> import org.apache.spark.streaming._  
import org.apache.spark.streaming._  
  
scala> import org.apache.spark.streaming.StreamingContext._  
import org.apache.spark.streaming.StreamingContext._  
  
scala>
```

Ahora

Colocamos la primera linea de codigo

```
val streamingContext: StreamingContext = new StreamingContext(sparkContext, Seconds(20))
```

1. Creamos variable StreamingContext, con un tiempo de analisis de datos de 20 segundos

```
scala> val ssc = new StreamingContext(sc, Seconds(20))  
ssc: org.apache.spark.streaming.StreamingContext = org.apache.spark.streaming.StreamingContext@18596d47
```

```
val lines: ReceiverInputStream[String] = streamingContext.socketTextStream("localhost", 9999)
```

2. Creamos un DStream que represente la transmission de datos desde una Fuente tcp especificada en el localhost y Puerto 9999

```
scala> val lines = ssc.socketTextStream("localhost", 9999)
lines: org.apache.spark.streaming.dstream.ReceiverInputDStream[String] = org.apache.spark.streaming.dstream.SocketInputDStream@6a8129d1
scala>
```

\$ nc -lk 9999

hello world

...

```
$ ./bin/run-example streaming.NetworkWordCount localhost 9999
```

...

Time: 1357008430000 ms

...

```
(hello, 1)
```

(world, 1)

...

```
val streamingContext: StreamingContext = new StreamingContext(sparkContext, Seconds(20))
val lines: ReceiverInputDStream[String] = streamingContext.socketTextStream("localhost", 9999)
```

RDD

$$\frac{1}{\sqrt{\pi}} \left(\frac{1}{\sqrt{\pi}} \int_0^x \frac{1}{t} dt \right) = \frac{1}{\pi} \ln x$$

version 3.3.1

```
Using Scala version 2.12.15 (OpenJDK 64-Bit Server VM, Java 11.0.17)
Type in expressions to have them evaluated.
Type :help for more information.
```

```
scala> val cadenas = Array("Docentes", "inteligenciaArtificial", "quefinal")
cadenas: Array[String] = Array(Docentes, inteligenciaArtificial, quefinal)
```

```
scala> val cadenasRDD = sc.parallelize (cadenas)
cadenasRDD: org.apache.spark.rdd.RDD[String] = ParallelCollectionRDD[0] at parallelize at <console>:24
```

```
scala> cadenasRDD.collect()
res0: Array[String] = Array(Docentes, inteligenciaArtificial, quefinal)
```

```
scala> val file = sc.textFile("/home/vmuser/textoRDD", 6)
file: org.apache.spark.rdd.RDD[String] = /home/vmuser/textoRDD MapPartitionsRDD[2] at textFile at <console>:23

scala> file.collect()
org.apache.hadoop.mapred.InvalidInputException: Input path does not exist: file:/home/vmuser/textoRDD
    at org.apache.hadoop.mapred.FileInputFormat.singleThreadedListStatus(FileInputFormat.java:304)
    at org.apache.hadoop.mapred.FileInputFormat.listStatus(FileInputFormat.java:244)
    at org.apache.hadoop.mapred.FileInputFormat.getSplits(FileInputFormat.java:332)
    at org.apache.spark.rdd.HadoopRDD.getPartitions(HadoopRDD.scala:208)
    at org.apache.spark.rdd.RDD.$anonfun$partitions$2(RDD.scala:292)
    at scala.Option.getOrElse(Option.scala:189)
    at org.apache.spark.rdd.RDD.partitions(RDD.scala:288)
    at org.apache.spark.rdd.MapPartitionsRDD.getPartitions(MapPartitionsRDD.scala:49)
    at org.apache.spark.rdd.RDD.$anonfun$partitions$2(RDD.scala:292)
    at scala.Option.getOrElse(Option.scala:189)
    at org.apache.spark.rdd.RDD.partitions(RDD.scala:288)
    at org.apache.spark.SparkContext.runJob(SparkContext.scala:2293)
    at org.apache.spark.rdd.RDD.$anonfun$collect$1(RDD.scala:1021)
    at org.apache.spark.rdd.RDDOperationScope$.withScope(RDDOperationScope.scala:151)
    at org.apache.spark.rdd.RDDOperationScope$.withScope(RDDOperationScope.scala:112)
    at org.apache.spark.rdd.RDD.withScope(RDD.scala:406)
    at org.apache.spark.rdd.RDD.collect(RDD.scala:1020)
    ... 47 elided
Caused by: java.io.IOException: Input path does not exist: file:/home/vmuser/textoRDD
    at org.apache.hadoop.mapred.FileInputFormat.singleThreadedListStatus(FileInputFormat.java:278)
    ... 63 more

scala> █
```

```
scala> val filtro = cadenasRDD.filter(line => line.contains("quefinal"))
filtro: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[3] at filter at <console>:23

scala> val fileNotFound = sc.textFile("/7añldlsjd/alkls/", 6)
fileNotFound: org.apache.spark.rdd.RDD[String] = /7añldlsjd/alkls/ MapPartitionsRDD[5] at textFile at <console>:23

scala> fileNotFound.collect()
org.apache.hadoop.mapred.InvalidInputException: Input path does not exist: file:/7añldlsjd/alkls
    at org.apache.hadoop.mapred.FileInputFormat.singleThreadedListStatus(FileInputFormat.java:304)
    at org.apache.hadoop.mapred.FileInputFormat.listStatus(FileInputFormat.java:244)
    at org.apache.hadoop.mapred.FileInputFormat.getSplits(FileInputFormat.java:332)
    at org.apache.spark.rdd.HadoopRDD.getPartitions(HadoopRDD.scala:208)
    at org.apache.spark.rdd.RDD.$anonfun$partitions$2(RDD.scala:292)
    at scala.Option.getOrElse(Option.scala:189)
    at org.apache.spark.rdd.RDD.partitions(RDD.scala:288)
    at org.apache.spark.rdd.MapPartitionsRDD.getPartitions(MapPartitionsRDD.scala:49)
    at org.apache.spark.rdd.RDD.$anonfun$partitions$2(RDD.scala:292)
    at scala.Option.getOrElse(Option.scala:189)
    at org.apache.spark.rdd.RDD.partitions(RDD.scala:288)
    at org.apache.spark.SparkContext.runJob(SparkContext.scala:2293)
    at org.apache.spark.rdd.RDD.$anonfun$collect$1(RDD.scala:1021)
    at org.apache.spark.rdd.RDDOperationScope$.withScope(RDDOperationScope.scala:151)
    at org.apache.spark.rdd.RDDOperationScope$.withScope(RDDOperationScope.scala:112)
    at org.apache.spark.rdd.RDD.withScope(RDD.scala:406)
    at org.apache.spark.rdd.RDD.collect(RDD.scala:1020)
    ... 47 elided
Caused by: java.io.IOException: Input path does not exist: file:/7añldlsjd/alkls
    at org.apache.hadoop.mapred.FileInputFormat.singleThreadedListStatus(FileInputFormat.java:278)
    ... 63 more

scala> █
```

```
val cadenas = Array("Docentes", "inteligenciaArtificial", "quefinal")
val cadenasRDD = sc.parallelize(cadenas)
cadenasRDD.collect()
file.collect()
val filtro = cadenasRDD.filter(line => line.contains("quefinal"))
val fileNotFound = sc.textFile("/7añldlsjd/alkls/", 6)
fileNotFound.collect()
```

En github tienen que subir en un repositorio los códigos de cada pregunta(carpeta), darle mínimamente acceso a msilva@fcpn.edu.bo, [mandar al correo](#) con referencia "2o parcial 319", notificar al mismo correo hasta el día 12 de diciembre a horas 12:00.