SolidQ

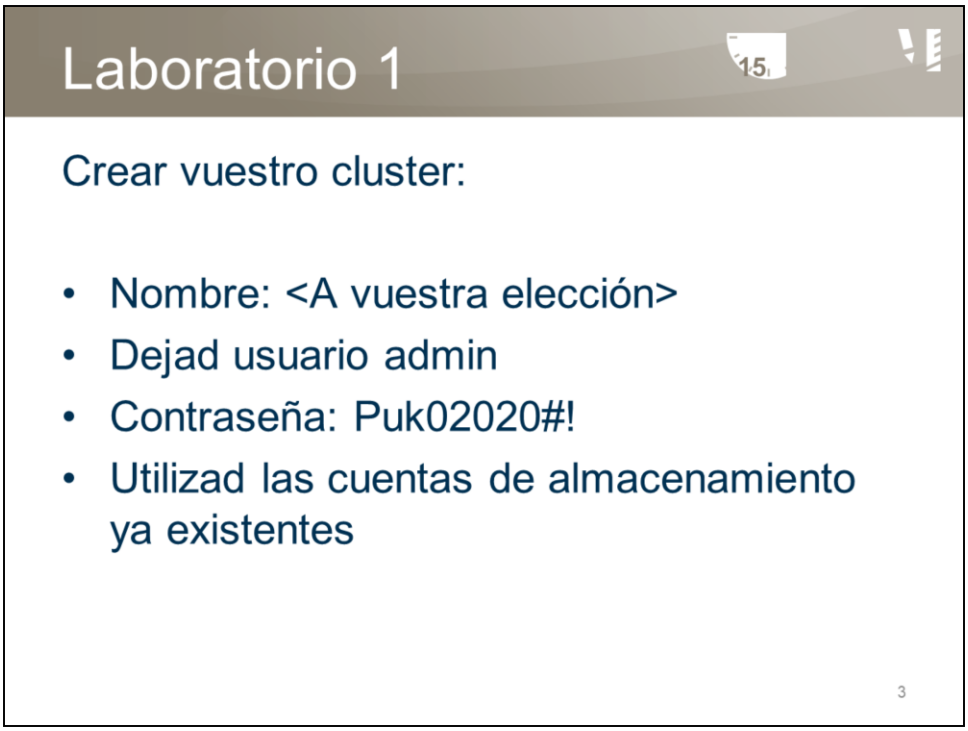
# BIG DATA: Diseño y Arquitectura de Soluciones con Hadoop, Spark y R

Antonio Soto
CEO
asoto@solidq.com

Vitae

# Laboratorios

- Laboratorio 1: Creación de un cluster HDInsight
- Laboratorio 2: HDFS
- Laboratorio 3: HIVE
- Laboratorio 4: SPARK
- Laboratorio 5: R




En este laboratorio se creará un cluster Hadoop desde el portal de Azure y se verá las opciones para interactuar con el cluster.

**Ejercicio 1: Crear el cluster**

- Conecta al portal azure https://portal.azure.com
- Inicia sesión con el usuario que se te haya asignado
- Arriba a la izquierda tienes la opción *Crear un Recurso*, selecciónala
- En el cuadro de buscar, pon HDInsight. Selecciona entre los resultados la opción ***HDInisght***, y dale a *Crear*
- En *Nombre del cluster* pon tu nombre de usuario
- En *Tipo de cluster* selecciona Hadoop y dale a Seleccionar
- En contraseña, pon la contraseña asignada
- En grupo de recursos pon Usar existente y selecciona el que aparece, dale a *Siguiente*
- En Seleccionar una cuenta de storage selecciona la que aparece y dale a siguiente. Se presentará el resumen en el que puedes darle a Crear para crear el cluster. Fíjate que podrías descargar una plantilla de despliegue

para poder desplegarlo de forma automatizada.

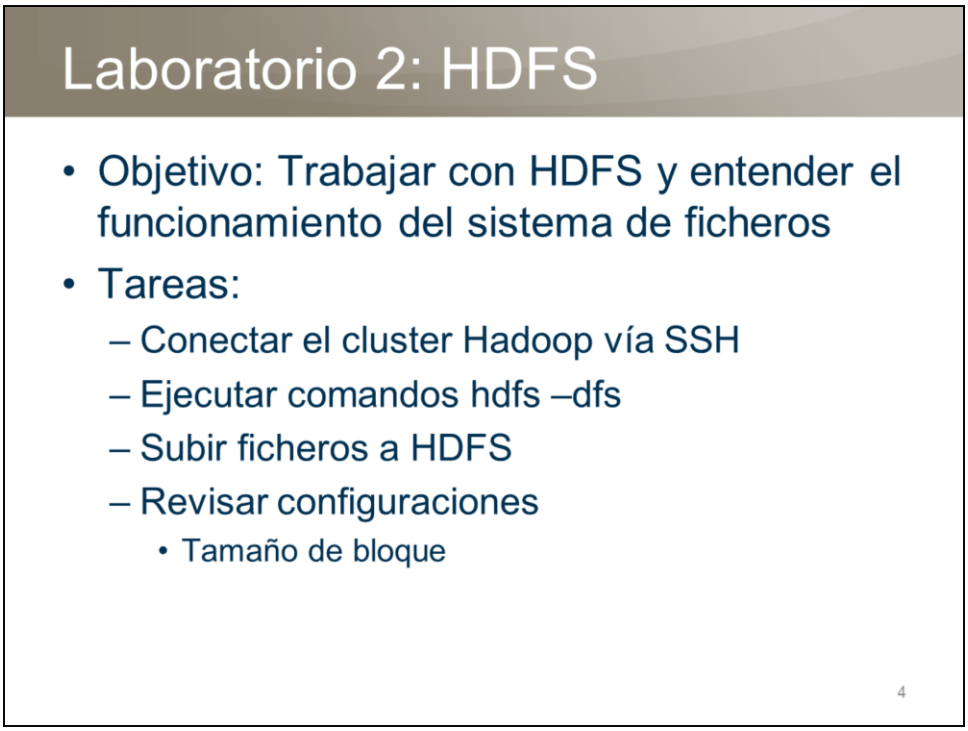
**Ejercicio 2: Entender el entorno**

Una vez desplegado el cluster, vamos a ver que opciones tenemos en el entorno Azure y posteriormente en nuestro cluster ya creado. En el menú de la izquierda busca *Todos los recursos* y busca tu cluster de HDInsight y hazle clic. SE abrirán las opciones de administración del recurso Azure. Podemos ver información sobre los nodos del cluster, cores disponibles, etc.

- En la parte izquierda buscamos la opción ***SSH e inicio de sesión del cluster***. Seleccionamos el nodo que nos aparece y copiamos la información de conexión.
- Desde un cliente SSH conectamos con esas credenciales.

**Ejercicio 3: Configurando nuestro equipo para acceso SSH y Tunnel**

Sigue las instrucciones del documento https://docs.microsoft.com/en-us/azure/hdinsight/hdinsight-linux-ambari-ssh-tunnel para configurar un túnel SSH y poder acceder a todas las opciones de administración del portal Ambari




En este segundo laboratorio, explorarás las opciones disponibles en HDFS para gestionar ficheros y el almacenamiento HDFS.

**Ejercicio 1: El Comando hdfs dfs**

Familiarízate con el comando hdfs dfs:

- Ejecuta hdfs dfs y revisa las opciones
- Ejecuta hdfs dfs –help ls
- Crea un directorio llamado data y comprueba que se ha creado
- Explora donde se encuentra realmente el almacenamiento. Vete a /Hadoop/hdfs/namenode

**Ejercicio 2: Copiando ficheros**

Al trabajar con una estructura “virtual” HDFS es necesario copiar las ficheros desde su almacenamiento original al HDFS . Para ello utilizaremos el –copyFromLocal:

1. Comprobamos que existe un fichero .log en /home/hive

    1. hdfs dfs –ls /home/hive
1. Copiamos el fichero
    1. hdfs dfs –copyFromLocal /home/hive/*.* /example/
2. Comprobamos que se ha copiado
    1. hdfs dfs –ls /example

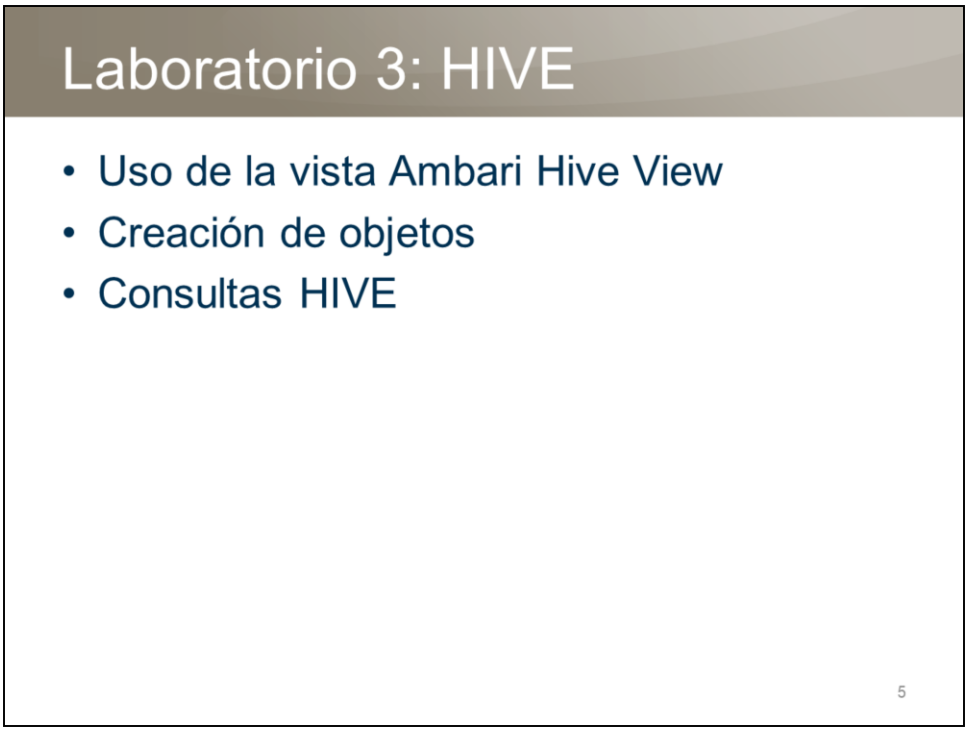
**Ejercicio 3: Generando HAR**

Con lo visto durante el módulo, genera un archivo HAR

**Ejercicio 4: Revisando configuración**

**Ejercicio 5: Aprovechando Azure**

- En el portal Azure busca la cuenta de almacenamiento
- Vete a blob
- Ahí verás la estructura HDFS
- Carga el fichero weblogs.csv en una carpeta llamada data




En este laboratorio de HIVE veremos como interactuar con el entorno para crear objetos, consultar objetos etc.

Para esta laboratorio es necesario que el archivo weblogs.csv esté en el directorio data

**Ejercicio 1: Las bases de HIVE**

**** *Es necesario cargar previamente el fichero weblogs.csv en la carpeta /data/weblogs de la infraestructura HDFS* ****

Realizaremos este laboratorio desde la Vista HIVE de Ambari, que nos permite interactuar gráficamente con el entorno. Conéctate y lanza la vista de usuario. Desde ahí ejecuta:

```
DROP DATABASE IF EXISTS HDILABDB CASCADE;
CREATE DATABASE HDILABDB;
Use HDILABDB;
CREATE EXTERNAL TABLE IF NOT EXISTS weblogs(
```

```
    TransactionDate varchar(50) ,
    CustomerId varchar(50) ,
    BookId varchar(50) ,
    PurchaseType varchar(50) ,
    TransactionId varchar(50) ,
    OrderId varchar(50) ,
    BookName varchar(50) ,
    CategoryName varchar(50) ,
    Quantity varchar(50) ,
    ShippingAmount varchar(50) ,
    InvoiceNumber varchar(50) ,
    InvoiceStatus varchar(50) ,
    PaymentAmount varchar(50)
) ROW FORMAT DELIMITED FIELDS TERMINATED by ',' lines
TERMINATED by '\n'
STORED AS TEXTFILE LOCATION '/data/weblogs';
```

Con esto, tendríamos una tabla externa. Comprueba que el fichero continúa en su ubicación original, y que puedes consultar el contenido de la tabla weblogs.

Ahora:

- Elimina la tabla
- Créala de nuevo, pero como tabla interna
- Carga el fichero con el comando:

```
LOAD DATA INPATH '/data/weblogs/weblogs.csv' INTO TABLE
HDILABDB.weblogs
```

Lo que creará la tabla weblogs en la base de datos HDILABDB. Puedes comprobar que se ha creado ejecutando:

```
hdfs dfs -ls /hive/warehouse
```

**Ejercicio 2: Algunas consultas**

Ejecuta las siguientes consultas:

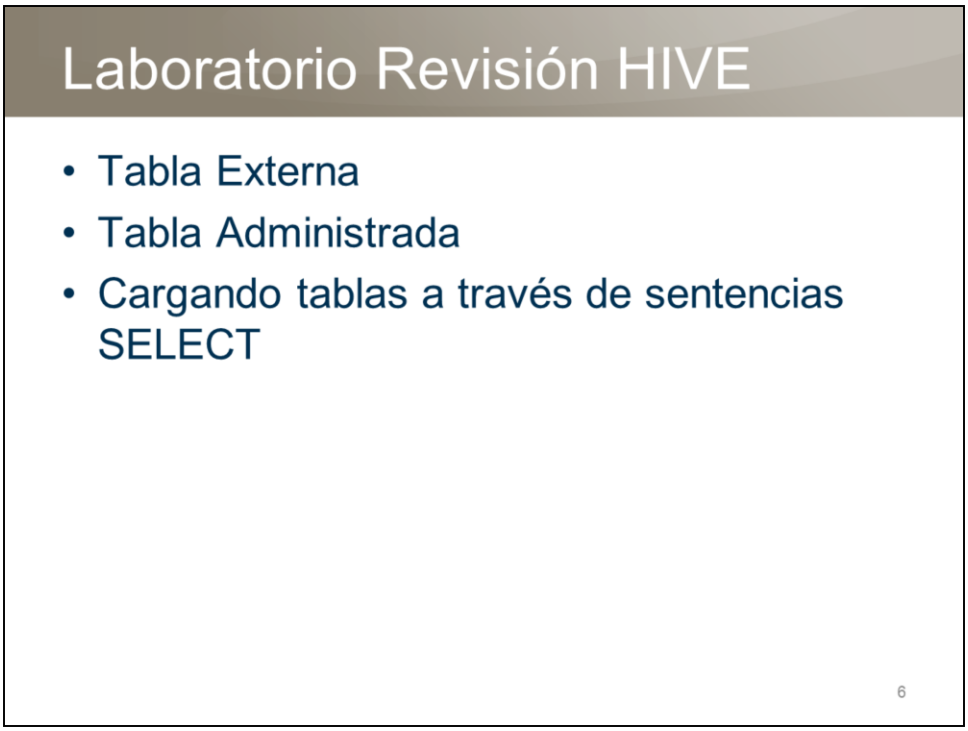
```
SELECT COUNT(*) FROM HDILABDB.weblogs;

SELECT * FROM HDILABDB.weblogs LIMIT 5;

SELECT * FROM HDILABDB.weblogs WHERE orderid='107';
```

Crea un nuevo *Worksheet* y ejecuta:

```
SELECT DISTINCT bookname FROM HDILABDB.weblogs WHERE
orderid='107';

SELECT bookname,COUNT(*) FROM HDILABDB.weblogs GROUP BY
bookname;
```

**Ejercicio 3: Problemas**

1. Ejecuta una consulta que devuelva el total pagado para cada categoría por mes
2. Ejecuta una consulta que devuelva la cantidad total pagada y la cantidad vendida de cada libro
3. Escribe una consulta que devuelva el top 3 de libros vistos por los usuarios que también visitaron **THE BOOK OF WITNESSES**




En este laboratorio vamos a repasar los conceptos básicos de HIVE. Para ello utilizaremos el fichero de ejemplo del directorio /HdiSamples/HdiSamples/WebsiteLogSampleData/SampleLog.

**Ejercicio 1: Tabla Externa**

- Revisamos desde la consola el contenido del directorio con el comando hdfs dfs –ls /HdiSamples/HdiSamples/WebsiteLogSampleData/SampleLog
- Revisamos el contenido del almacén de HIVE con el comando hdfs dfs –ls /hive/warehouse
- Creamos la tabla externa a través del comando:

```
CREATE EXTERNAL TABLE samplelog(
  fecha date,
  time varchar(200),
  sitename varchar(200),
  method varchar(200),
  uristem varchar(200),
```

```
  uriquery varchar(200),
  port varchar(200),
  username varchar(200),
  ip varchar(200),
  UserAgent varchar(200),
  Cookie varchar(200),
  Referer varchar(200),
  host varchar(200),
  status varchar(200),
  substatus varchar(200),
  win32substatus varchar(200),
  scbytes int,
  csbytes int,
  timetaken int)
  ROW FORMAT DELIMITED FIELDS TERMINATED BY ' '
STORED AS TEXTFILE LOCATION
'/HdiSamples/HdiSamples/WebsiteLogSampleData/SampleLog'
tblproperties ("skip.header.line.count"="2");
```

- Ejecutamos SELECT * FROM samplelog; y obtenemos el resultado deseado
- Ejecutamos SELECT COUNT(*) from samplelog; y anotamos el número de filas
- Copiamos el fichero de origen con el comando hdfs dfs -cp /HdiSamples/HdiSamples/WebsiteLogSampleData/SampleLog/909f2b.log /HdiSamples/HdiSamples/WebsiteLogSampleData/SampleLog/909f2c.log
- Ejecutamos de nuevo SELECT COUNT(*) from samplelog; y obtendremos el doble de filas
- Si volvemos a consultar el contenido de los directorios, tanto del almacen de hive, como el origen, nada ha cambiado
- Eliminamos el fichero que hemos copiado /HdiSamples/HdiSamples/WebsiteLogSampleData/SampleLog/909f2c.log
- Ejecutamos de nuevo SELECT (COUNT*) from samplelog; y ha desaparecido el contenido de ese fichero

- Eliminamos la tabla externa DROP TABLE samplelog;

Como hemos podido observar la tabla externa no realiza ninguna operación con los datos origen. Tan solo muestra en formato tabla el contenido de un directorio. Los datos se “cargan” en el momento en el que se consultan, aplicando las “transformaciones” que se hayan definido en la consulta CREATE TABLE

**Ejercicio 2: Tabla Administrada**

Vamos a crear ahora una tabla administrada sobre los mismos datos de ejemplo. Para ello ejecutamos el comando:

```
CREATE TABLE samplelogadmin(
  fecha date,
  time varchar(200),
  sitename varchar(200),
  method varchar(200),
  uristem varchar(200),
  uriquery varchar(200),
  port varchar(200),
  username varchar(200),
  ip varchar(200),
  UserAgent varchar(200),
  Cookie varchar(200),
  Referer varchar(200),
  host varchar(200),
  status varchar(200),
  substatus varchar(200),
  win32substatus varchar(200),
  scbytes int,
  csbytes int,
  timetaken int)
```

ROW FORMAT DELIMITED FIELDS TERMINATED BY ' '
tblproperties ("skip.header.line.count"="2");

- Consultamos el contenido de la tabla y vemos que está vacía
- Ejecutamos: LOAD DATA INPATH '/HdiSamples/HdiSamples/WebsiteLogSampleData/SampleLog/909f2b.log' INTO TABLE samplelogadmin
- Consultamos los directorios de origen y fichero y de almacén de Hive (hive/warehouse) ¿Qué ha ocurrido?

**Ejercicio 3: Cargando tablas desde sentencias SELECT**

En este ejercicio crearemos dos tablas externas sobre las que después generaremos otras dos tablas con contenido agregado que cargaremos a través de sentencias CREATE TABLE----- SELECT------

Ejecuta el siguiente script:

DROP TABLE IF EXISTS hvac;

--crear la tabla externa hvac sobre el csv
CREATE EXTERNAL TABLE hvac(dates STRING, time STRING, targettemp BIGINT,
actualtemp BIGINT, system BIGINT, systemage BIGINT, buildingid BIGINT)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
STORED AS TEXTFILE LOCATION '/HdiSamples/HdiSamples/SensorSampleData/hvac/';

DROP TABLE IF EXISTS building;

--crear la tabla externa building sobre el csv
CREATE EXTERNAL TABLE building(buildingid BIGINT, buildingmgr STRING,
buildingage BIGINT, hvacproduct

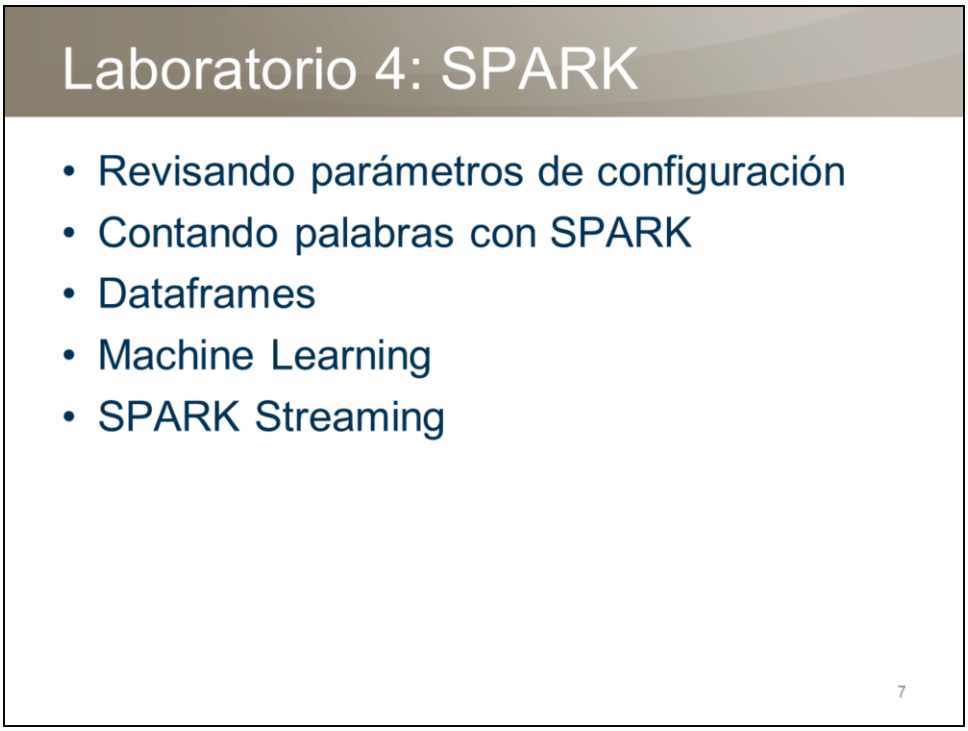
```
STRING, country STRING)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
STORED AS TEXTFILE LOCATION
'/HdiSamples/HdiSamples/SensorSampleData/building/';

DROP TABLE IF EXISTS hvac_temperatures;

--crear la tabla administrada hvac_temperatures desde la tabla hvac. Mostrar
donde se almacena esa tabla
CREATE TABLE hvac_temperatures AS
SELECT *, targettemp - actualtemp AS temp_diff,
        IF((targettemp - actualtemp) > 5, 'COLD',
        IF((targettemp - actualtemp) < -5, 'HOT', 'NORMAL')) AS
temprange,
        IF((targettemp - actualtemp) > 5, '1', IF((targettemp - actualtemp)
< -5, '1', 0)) AS extremetemp
FROM hvac;

DROP TABLE IF EXISTS hvac_building;

-- crear la tabla hvac_building haciendo join entre las tablas de building y
hvac_temperatures
CREATE TABLE hvac_building AS
SELECT h.*, b.country, b.hvacproduct, b.buildingage, b.buildingmgr
FROM building b JOIN hvac_temperatures h ON b.buildingid = h.buildingid;
```

Comprueba que se han cargado datos en las dos últimas tablas a partir del resultado de las consultas SELECT. ¿Dónde se han almacenado?

Limpiamos el entorno:

```
DROP TABLE hvac;
DROP TABLE building;
```

```
DROP TABLE hvac_temperaturas;
DROP TABLE hvac_Building;
```




Antes de comenzar el ejercicio vamos a actualizar la versión de las funciones Python instaladas. Ejecuta el siguiente comando desde el Shell:

sudo -HE /usr/bin/anaconda/bin/conda install pandas

**Ejercicio 1: Revisando parámetros de configuración**

Los tres parámetros clave que se pueden utilizar para la configuración de Spark según los requisitos de la aplicación son **spark.executor.instances**, **spark.executor.cores** y **spark.executor.memory**. Un ejecutor es un proceso que se inicia para una aplicación Spark. Se ejecuta en el nodo de trabajo y es responsable de realizar las tareas de la aplicación. El número predeterminado de ejecutores y el tamaño de estos para cada clúster se calcula en función del número de nodos de trabajo y el tamaño de estos. Esta información se almacena en **spark-defaults.conf (/etc/spark2/2.6.2.25-1/0)** en los nodos principales del clúster. Los tres parámetros de configuración se pueden configurar en el nivel de clúster (para todas las aplicaciones que se ejecutan

en el clúster) o se pueden especificar también para cada aplicación individual.

Es habitual que dependiendo del trabajo a realizar estas configuraciones se modifican para un trabajo en concreto en tiempo de ejecución, en lugar de modificarlas para todo el cluster. Vamos a ver como hacerlo desde un Notebook de Jupyter. Para ello:

- Iniciamos el entorno de JUPYTER bien desde el portal de Azure o bien directamente en el navegador: https://NOMBRECLUSTER/ JUPYTER
- Creamos una nueva carpeta llamada Curso
- Dentro de la carpeta cargamos el fichero 00 Empezando con SPARK
- Insertamos una celda de tipo Code como primera celda y especificamos

```
%%configure
{"executorMemory": "3072M", "executorCores": 4, "numExecutors":10}
```

- Ejecutamos la celda y nos mostrará el cambio de configuración. Si quisiésemos cambiar la configuración una vez iniciada la sesión Spark, deberíamos de especificar el parámetro -f

**Ejercicio 2: Cuenta palabras con SPARK**

La segunda celda del Notebook nos muestra como cargar ficheros de texto y manipularlos desde Spark. El contexto que tenemos cargado de forma predeterminada utiliza YARN como modo de ejecución. Ejecutamos la segunda celda revisando las operaciones que va realizando en cada línea

**Ejercicio 3: DataFrames**

En este ejercicio, verás como trabajar con DataFrames importando datos desde ficheros, en este caso, csv. Verás también como aplicar transformaciones y filtros sobre esos datos, así como la creación de estructuras en memoria que te permitan después utilizar sentencias SELECT para la consulta de esos datos. Finalizarás el ejercicio viendo la interacción entre SPARKSQL y HIVE, almacenando los datos en una tabla y consultando tablas HIVE desde SPARK. Vete ejecutando las celdas desde la cabecera de DataFrames.

**Ejercicio 4: Machine Learning**

En este laboratorio veremos como crear, entrenar y consultar un modelo predictivo con Spark

Los datos siguientes muestran la temperatura objetivo y la temperatura real de algunos edificios con sistemas HVAC instalados. La columna System representa el identificador del sistema y la columna SystemAge, el número de años que lleva el sistema HVAC instalado en el edificio. Los datos de este tutorial se usan para predecir si un edificio será más cálido o frío en función de la temperatura objetivo, dados un identificador del sistema y la antigüedad del sistema

Cargaremos los datos del .csv. A través de una función que compara la temperatura real con la temperatura de destino indicaremos que si la temperatura real es mayor, el edificio está **cálido**, lo que viene indicado por el valor 1.0. De lo contrario, el edificio está **frío**, lo que se indica con el valor 0.0.

Generaremos un **pipeline** de Spark, utilizando un algoritmo de Regresión Logística.

Un **pipeline** es es un flujo de trabajo completo que combina varios algoritmos de ML. Los pipelines definen las fases y el orden de un proceso de ML.

Un **transformador** es un algoritmo que transforma un elemento DataFrame en otro. Por ejemplo, un transformador de características puede leer una columna de un elemento DataFrame, asignarla a otra columna y generar un nuevo elemento DataFrame con la columna asignada anexada

Un **estimador** es una abstracción de algoritmos de aprendizaje y es responsable del ajuste o el aprendizaje en un conjunto de datos para producir un transformado

En nuestro caso generaremos un pipeline que consta de tres fases: tokenizer, hashingTF (ambos son transformadores) y lr:

1. En la primera fase, Tokenizer, se divide la columna de entrada SystemInfo (que consta de los valores de identificador y antigüedad del sistema) en una columna de salida words. Esta nueva columna words se agrega a DataFrame
2. En la segunda fase, HashingTF, se convierte la nueva columna words en

vectores de característica. Esta nueva columna features se agrega al elemento DataFrame. Estas dos primeras fases son transformadores.

3. La tercera fase, LogisticRegression, es un estimador, por lo que la canalización llama al método LogisticRegression.fit() para generar un modelo LogisticRegressionModel.

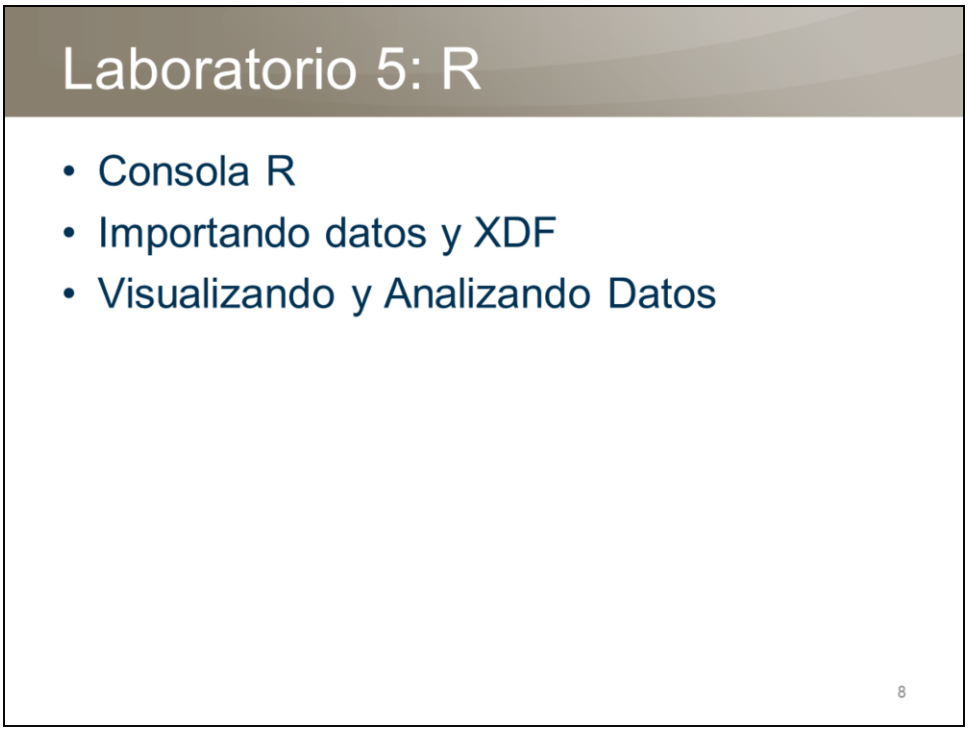
Prepararemos después unos datos de prueba para probar el modelo y veremos sus predicciones.

Para todo ello, vete paso a paso a través del laboratorio 01 SPARK ML HVAC

**Ejercicio 5: SPARK Streaming**

Veremos en este ejercicio las bases de SPARK Streaming para crear aplicaciones que sean capaces de analizar datos en tiempo real a través de streams. Carga para ello el notebook 02 Bases de Spark Streaming y ejecuta las instrucciones como se te indican en el Notebook

En un escenario más real, lo habitual es utilizar un streaming estructurado que acabe generando tablas o ficheros para un consumo posterior por parte de ETL o directamente herramientas de BI. Para ello revisa el Notebook 03 Spark Streaming Structurado




En este laboratorio veremos como trabajar con un entorno de R instalado en un cluster Spark sobre Hdinsight. Recuerda que una de las diferencias entre este tipo de cluster y los cluster Spark o HDInsight es que aquí tenemos un nodo Edge al que debemos de conectarnos para interactuar, en el que estará instalado el entorno R y desde el que se configura todo ese entorno.

**Ejercicio 1 Consola R**

- Conecta por SSH al nodo –ed de tu cluster
- Ejecuta el comando **R** para entrar en el entorno
- Ahora ya podrías ejecutar sentencias R en el servidor Edge, por ejemplo

```
getwd()
list.files(rxGetOption("sampleDataDir"))
```

Puedes consultar otras opciones como: numCoresToUse, blocksPerRead, computeContext(), spark.executorCores,…..

## Ejercicio 2 Importando datos a R y gestionando XDF

En este ejercicio veremos las diferentes opciones que tenemos para cargar datos en estructuras R, y gestionar los XDF generados aplicando diferentes técnicas. Utiliza el fichero 01 Importando Datos y XDF para seguir paso a paso el ejercicio

## Ejercicio 3 Visualizando y Analizando Datos

Una vez hemos visto como cargar y manejar los datos, el siguiente paso es dedicar un poco más de tiempo para visualizar y analizar esos datos, a través de una regresión logística para realizar una predicción. Sigue para ello el fichero 02 Visualizando y Analizando datos


SolidQ
www.solidq.com
info@solidq.com