

Minería de Datos II

Clase 10 - Apache Spark

Apache Spark es un motor para procesamiento a gran escala de datos, integrado, rápido, "in memory" y de propósito general.

Tienen su propio sistema de "Cluster Management" y utiliza Hadoop solo como almacenamiento.

Spark está escrito en Scala y provee APIs en Java, Scala, Python y R.

Características

- Computación en memoria.
- Tolerancia a fallos.
- Multipropósito.



Graph Processing



Machine Learning



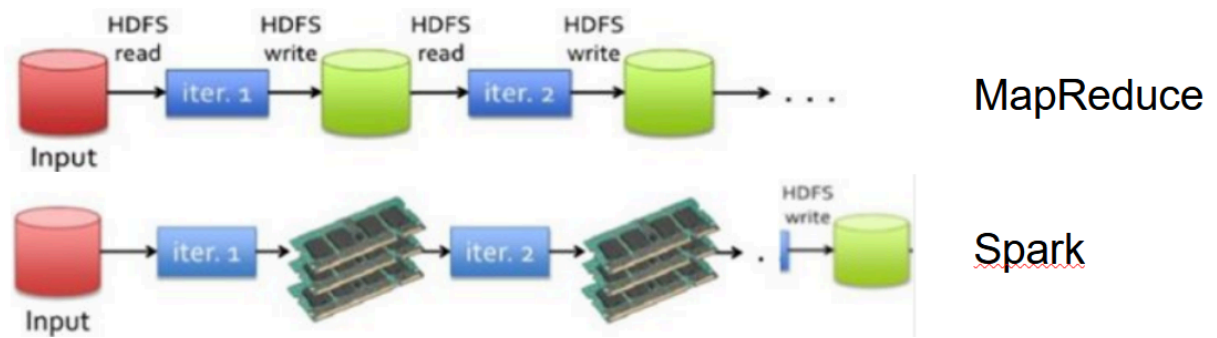
Streaming



Analysis

Spark es ideal para tareas de procesamiento iterativo e interactivo de grandes "data sets" y flujos de datos ("streaming").

Brinda una performance entre 10-100x mayor que Hadoop operando con construcciones de datos ("data constructs") llamadas "Resilient Distributed Datasets" (RDDs), esto ayuda a evitar latencias en tareas de lectura y escritura en discos. Es una alternativa a MapReduce.



-¿Cómo utilizar Spark?

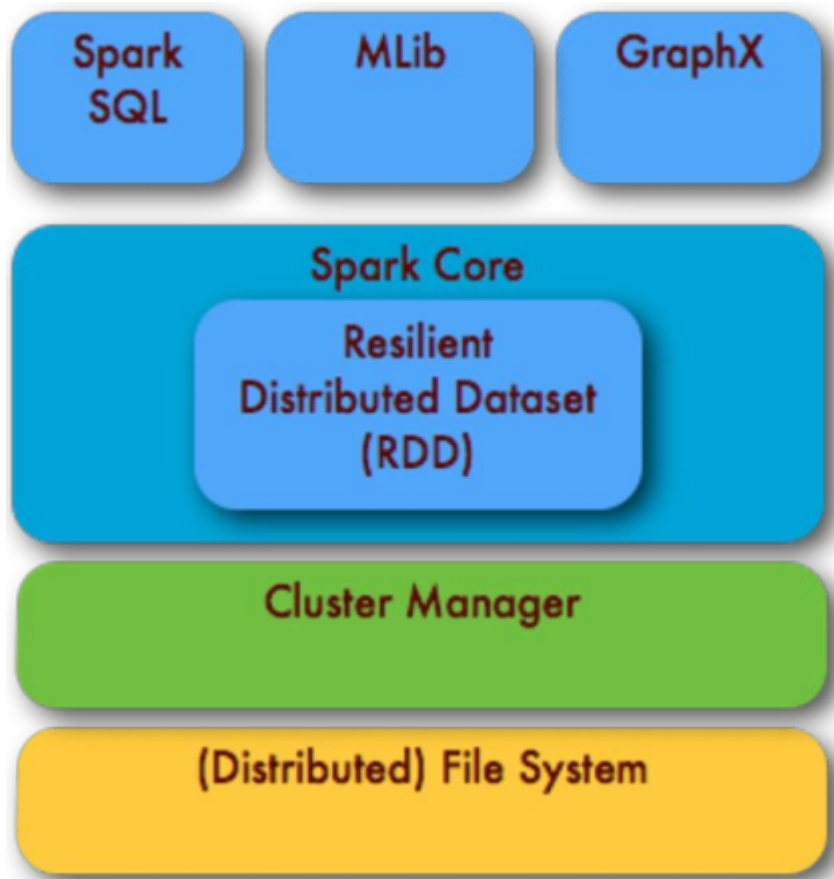


-Arquitectura

Spark tiene una arquitectura de capas bien definida donde todos los componentes están relacionados e integrados con extensiones y librerías.

Está basado en dos abstracciones:

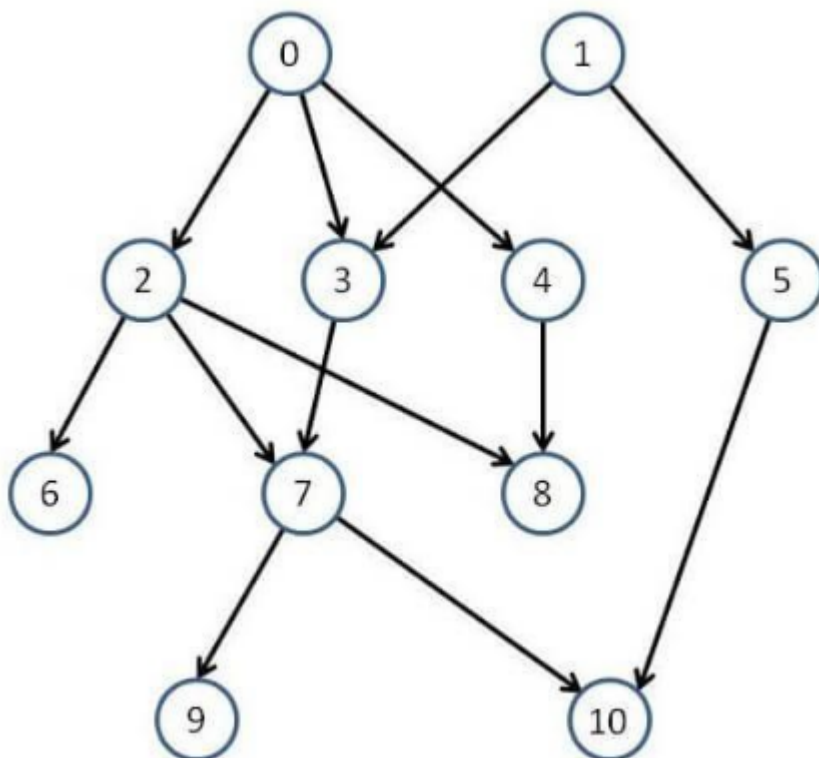
- RDD (Resilient Distributed Dataset)
- DAG (Directed Acyclic Graph)



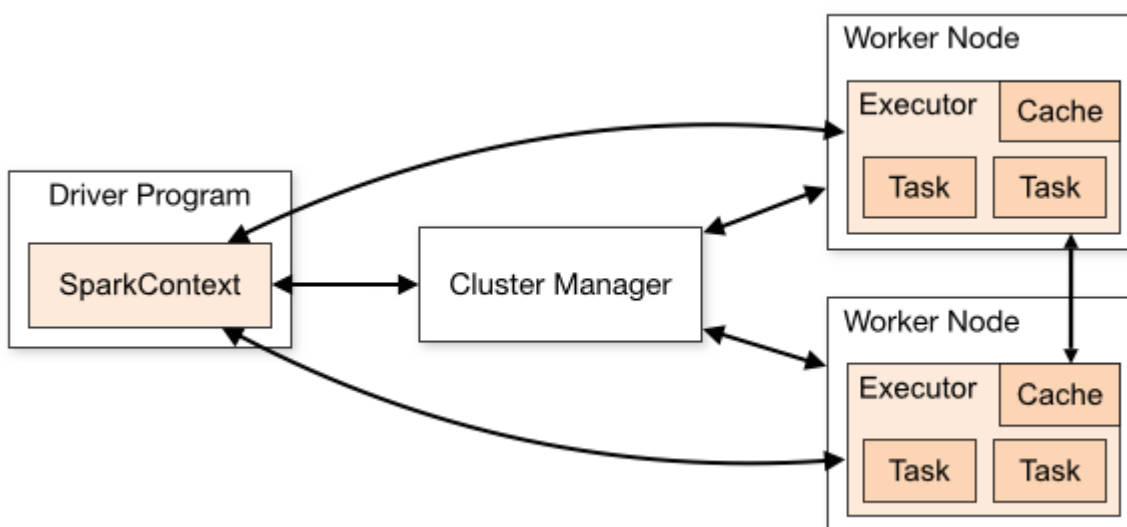
Cada tarea de Spark crea un DAG de etapas de trabajo para que se ejecuten en un determinado cluster.

En comparación con MapReduce, el cual crea un DAG con dos estados predefinidos (Map y Reduce), los grafos DAG creados por Spark pueden tener cualquier número de etapas. Spark con DAG es más rápido que MapReduce por el hecho de que no tiene que escribir en disco los resultados obtenidos en las etapas intermedias del grafo.

Gracias a una completa API, es posible programar complejos hilos de ejecución paralelos en unas pocas líneas de código.



Es un grafo dirigido que no tiene ciclos, es decir, para cada nodo del grafo no hay un camino directo que comience y finalice en dicho nodo. Un vértice se conecta a otro, pero nunca a sí mismo; para cada vértice v , no hay un camino directo que empiece y termine en v . Spark soporta el flujo de datos acíclico.



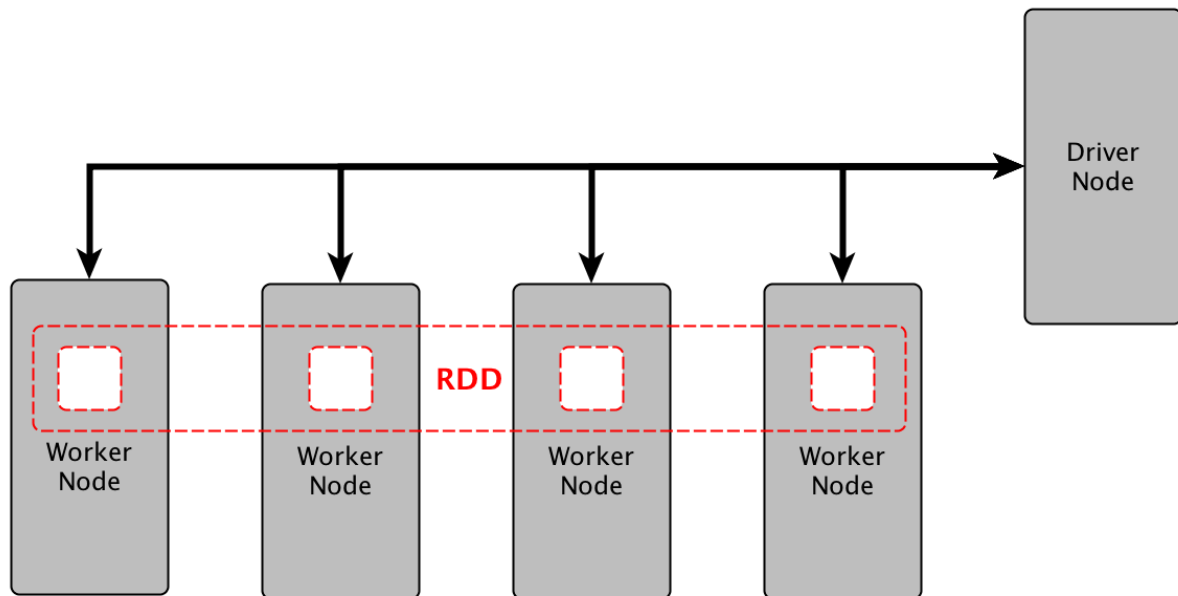
-RDD

Es la estructura fundamental de datos de Apache Spark, una colección de objetos que se computan en diferentes nodos del Cluster.

Resilient (tolerante a fallos), capacidad de recomponer particiones de datos dañadas o perdidas por fallos en nodos.

Distributed, los datos residen en varios nodos.

DataSet, representa registros de los datos, que el usuario puede cargar en forma de archivos JSON, CSV, texto o bases de datos por medio de JDBC sin una estructura de datos específica.



Todas las aplicaciones en Spark poseen un manejador central de programa (Driver) y varios ejecutores que se crean a lo largo del clúster, estas son las computadoras que realizarán las tareas en paralelo y finalmente devolverán los valores al driver, la aplicación central.

-Operaciones sobre RDD's

Para poder realizar estas tareas, Spark posee desde su versión 1.0 los RDD (Resilient Distributed Dataset), los cuales son tolerantes a fallos y pueden ser distribuidos a lo largo de los nodos del clúster.

Los RDD pueden ser creados al cargar datos de manera distribuida, como es desde un HDFS, Cassandra, Hbase o cualquier sistema de datos soportado por Hadoop, pero también por colecciones de datos de Scala o Python, además de poder ser leídos desde archivos en el sistema local.

En visión general, un RDD puede ser visto como un set de datos los cuales soportan solo dos tipos de operaciones: transformaciones y acciones.

Las transformaciones permiten crear un nuevo RDD a partir de uno previamente existente, mientras que las acciones retornan un valor al driver de la aplicación. El núcleo de operación del paradigma de Spark es la ejecución perezosa (Lazy), es decir que las transformaciones solo serán calculadas posterior a una llamada de acción.

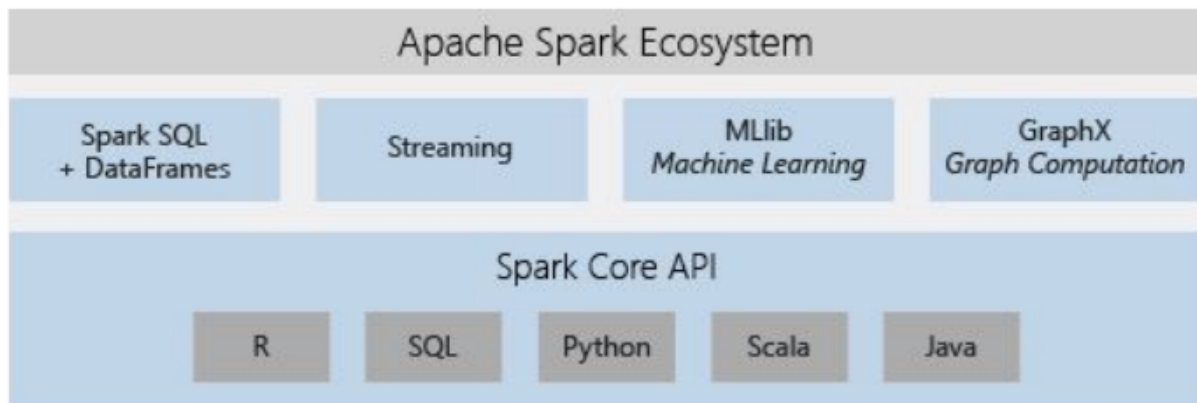
Además, los RDD poseen una familiaridad con el paradigma orientado a objetos, lo cual permite que podamos realizar operaciones de bajo nivel a modo. Map, filter y reduce son tres de las operaciones más comunes.

Una de las grandes ventajas que ofrecen los RDD es la compilación segura; por su particularidad de ejecución perezosa, se calcula si se generará un error o no antes de ejecutarse, lo cual permite identificar problemas antes de lanzar la aplicación. El pero que podemos encontrar con los RDD es que no son correctamente tratados por el Garbage collector y cuando las lógicas de operación se hacen complejas, su uso puede resultar poco práctico, aquí entran los DataFrames.

-Arquitectura Hadoop - Spark

| | Hadoop | Spark |
|-----------------------------------|---|---|
| Propósito | Procesamiento y almacenamiento de grandes datasets | Motor de propósito general para procesamiento de datos a gran escala |
| Componentes principales | Hadoop Distributed File System - HDFS | Spark Core, motor de procesamiento en memoria |
| Almacenamiento | HDFS administra colecciones de datos a través de múltiples nodos en un cluster de servidores "commodity" | Spark no realiza almacenamiento distribuido, opera en colecciones de datos distribuidas |
| Tolerancia a fallos | Replicación. Los datos son escritos a discos en varios nodos luego de cada operación | RDDs, minimizando "network I/O". Los RDDs pueden ser reconstruidos ante fallos |
| Velocidad de Procesamiento | MapReduce es más lento | Hasta 10x más rápido que MapReduce para "batch processing" y hasta 100x más rápido para "streaming processing" |
| Soporte de lenguajes | Java | Scala, APIs para Python, Java, R y otros |

-Módulos



Spark Core: Brinda velocidad dando capacidades de computación "in-memory". Spark Core es la base del procesamiento distribuido de grandes datasets.

- SparkSQL <https://spark.apache.org/sql/>
 - Lenguaje provisto para tratar con datos estructurados.
 - Permite ejecutar consultas sobre los datos y obtener resultados útiles.
 - Soporta consultas a través de SQL y HiveQL.
- Spark Streaming <https://spark.apache.org/streaming/>
 - Permite procesamiento de flujos en forma escalable, rápida y tolerante a fallos.
 - Spark divide los "streams" de datos en pequeños "batches".
 - Puede acceder a orígenes de datos como Kafka, Flume, Kinesis o sockets TCP.
 - Trata a cada "batch" de datos como RDDs y los procesa.
 - Puedo operar con varios algoritmos.
 - La data procesada se almacena en el file system y bases de datos.
- Spark MLlib <https://spark.apache.org/mllib/>
 - Es una librería escalable de Machine learning.
 - Contiene librerías para implementar algoritmos de ML, por ejemplo clustering, regression y Filtrado colaborativo.
 - El workflow ML incluye estandarización, normalización, hashing, algebra lineal, estadísticas.
- Spark GraphX <https://spark.apache.org/graphx/>
 - Es un motor de análisis de grafos de red y un almacén de datos.
 - Extiende Spark RDD brindando una abstracción gráfica de grafos dirigidos con propiedades asignadas a cada nodo y vértice.

Laboratorio:

Para ejecutar los ejercicios en los notebooks de Databricks vamos a necesitar un cluster

<https://docs.databricks.com/getting-started/quick-start.html#quick-start>

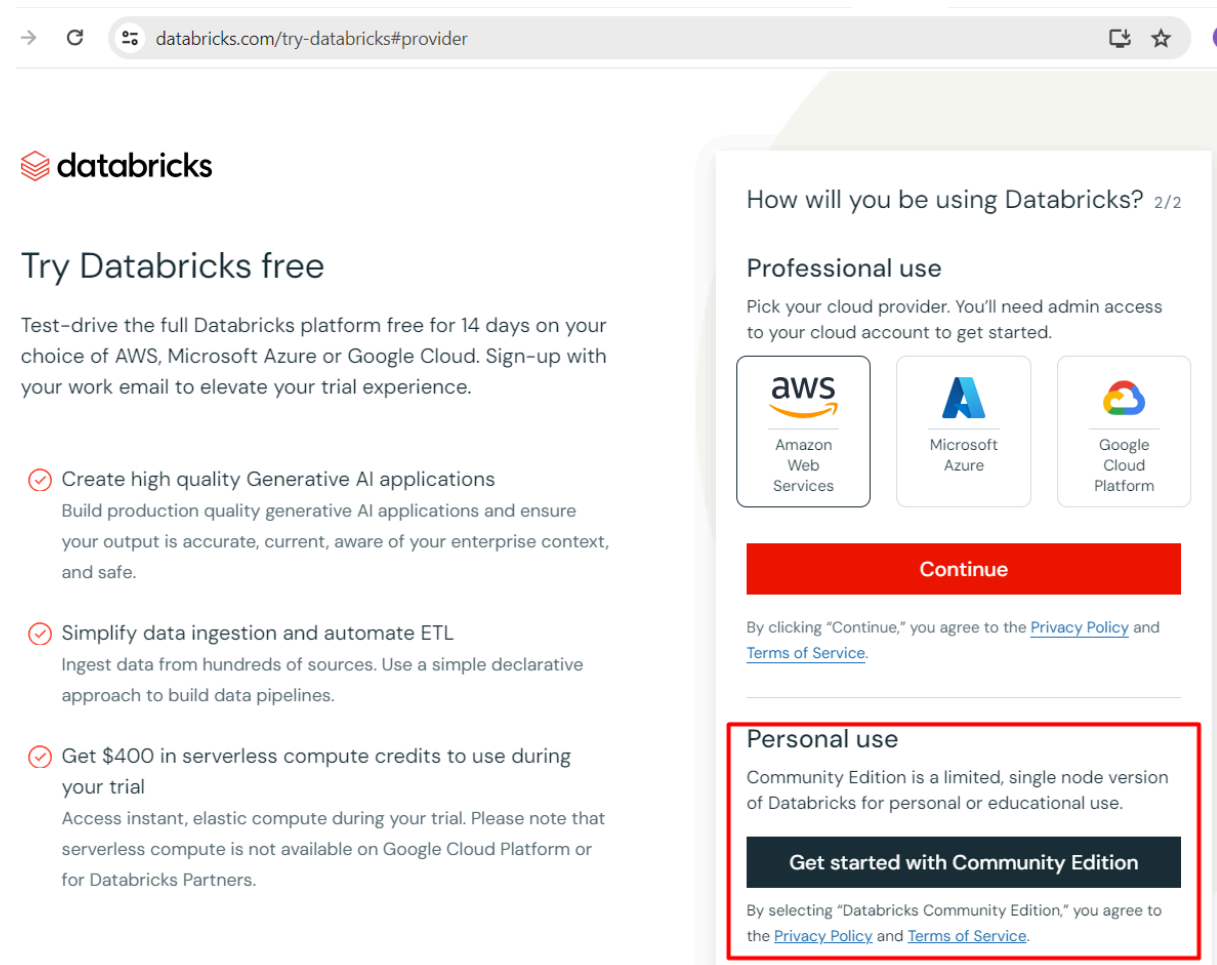
DataFrames (Python)

DataFrames (Dataframes.ipynb)

Dataset de Ventas de Supermercados (Demo-Ventas.ipynb)

Datasets (Scala)

https://docs.databricks.com/_static/notebooks/getting-started/iotdevicegeoipds.html



databricks

Try Databricks free


Test-drive the full Databricks platform free for 14 days on your choice of AWS, Microsoft Azure or Google Cloud. Sign-up with your work email to elevate your trial experience.


- ✓ Create high quality Generative AI applications
Build production quality generative AI applications and ensure your output is accurate, current, aware of your enterprise context, and safe.
- ✓ Simplify data ingestion and automate ETL
Ingest data from hundreds of sources. Use a simple declarative approach to build data pipelines.
- ✓ Get \$400 in serverless compute credits to use during your trial
Access instant, elastic compute during your trial. Please note that serverless compute is not available on Google Cloud Platform or for Databricks Partners.


How will you be using Databricks? 2/2

Professional use

Pick your cloud provider. You'll need admin access to your cloud account to get started.


Amazon Web Services


Microsoft Azure


Google Cloud Platform

Continue

By clicking "Continue," you agree to the [Privacy Policy](#) and [Terms of Service](#).

Personal use

Community Edition is a limited, single node version of Databricks for personal or educational use.

Get started with Community Edition

By selecting "Databricks Community Edition," you agree to the [Privacy Policy](#) and [Terms of Service](#).

Importante! "Get Started with Community Edition"

1-Crear el Cluster

Home

Workspace

Recents

Data

Clusters

Jobs

Create Cluster

New Cluster

Cancel

Create Cluster

0 Workers: 0.0 GB Memory, 0 Cores, 0 DBU
1 Driver: 15.3 GB Memory, 2 Cores, 1 DBU

Cluster Name

Training

Databricks Runtime Version

Runtime: 6.5 (Scala 2.11, Spark 2.4.5)

New This Runtime version supports only Python 3.

Instance

Free 15GB Memory: As a Community Edition user, your cluster will automatically terminate after an idle period of two hours. For [more configuration options](#), please [upgrade your Databricks subscription](#).

Instances

Spark

Availability Zone

us-west-2c

2-Importar Notebook

Home

Workspace

Recents

Data

Workspace

Workspace

Shared

Users

Educacion-IT

Create

Import

Permissions

Sort

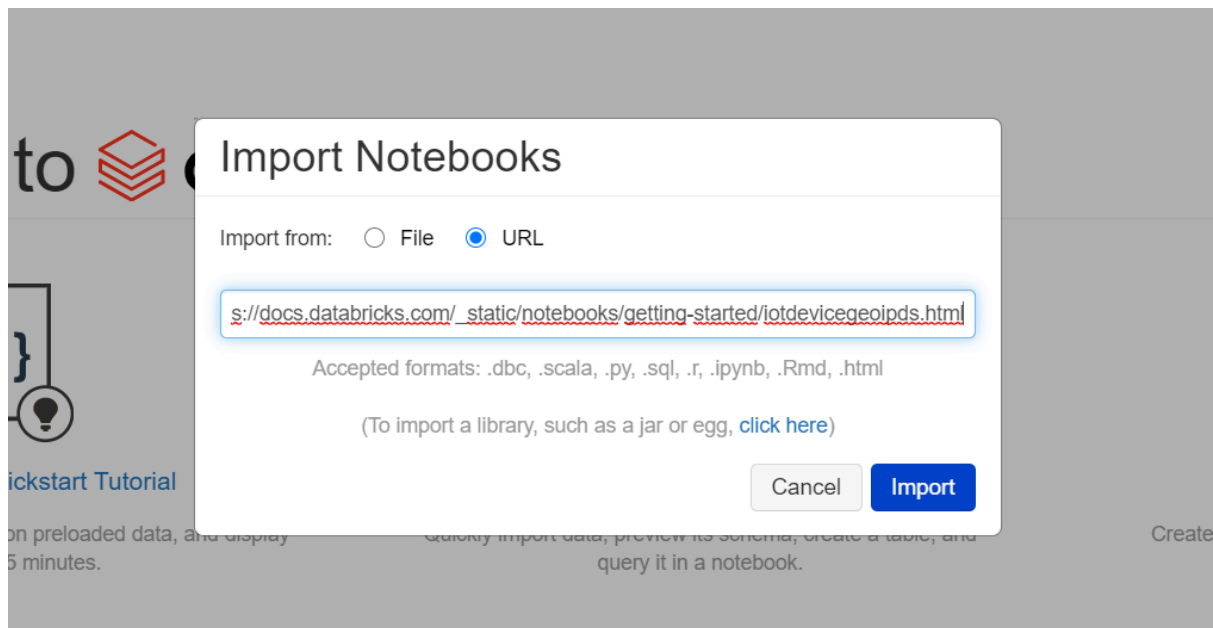
to databricks

Drop files or [click to browse](#)

[Import & Explore Data](#)

Quickly import data, preview its schema, create a table, and query it in a notebook.

3-Ingresa URL



4-Adjuntar Clúster y Ejecutar

Process IoT Device Data Using Datasets (Scala)

Training

```
//read the JSON file and create the dataset from the case class DeviceIoTData
// ds is now a collection of JVM Scala objects DeviceIoTData
val ds = spark.read.json(s"/databricks-datasets/iot/iot_devices.json").as[DeviceIoTData]
```

(1) Spark Jobs

ds: org.apache.spark.sql.Dataset[DeviceIoTData]

```
battery_level: long
c02_level: long
cca2: string
cca3: string
cn: string
device_id: long
device_name: string
humidity: long
ip: string
latitude: double
lcd: string
longitude: double
scale: string
temp: long
timestamp: long
```

ds: org.apache.spark.sql.Dataset[DeviceIoTData] = [battery_level: bigint, c02_level: bigint ... 13]