

# Big Data Processing

## — L06: Gentle Introduction to Apache Spark —

---

---

**Dr. Ignacio Castineiras**  
Department of Computer Science

# Outline

1. Introduction to Apache Spark.
2. A Spark Application: General Overview.
3. Two Different Roles: IT Manager vs. Data Engineer.
4. Databricks: An Online Platform for Data Engineers.

# Outline

1. Introduction to Apache Spark.
2. A Spark Application: General Overview.
3. Two Different Roles: IT Manager vs. Data Engineer.
4. Databricks: An Online Platform for Data Engineers.

# Introduction to Apache Spark

Apache Spark is an



- open-source
- distributed
- general-purpose
- cluster-computing

framework designed for 3 purposes:

# Introduction to Apache Spark

Apache Spark is an



- open-source
- distributed
- general-purpose
- cluster-computing

framework designed for 3 purposes:

1. Be easy to use - you can develop applications locally, using a high-level API letting you focus on the content of your computation.

# Introduction to Apache Spark

Apache Spark is an



- open-source
- distributed
- general-purpose
- cluster-computing

framework designed for 3 purposes:

1. Be easy to use - you can develop applications locally, using a high-level API letting you focus on the content of your computation.
2. Be fast - enabling interactive use and complex algorithms.

# Introduction to Apache Spark

Apache Spark is an



- open-source
- distributed
- general-purpose
- cluster-computing

framework designed for 3 purposes:

1. Be easy to use - you can develop applications locally, using a high-level API letting you focus on the content of your computation.
2. Be fast - enabling interactive use and complex algorithms.
3. Be general - allowing to combine multiple types of computations, including text, SQL, graph and machine learning processing (both offline and online) that might previously have required different engines.

# Introduction to Apache Spark

Apache Spark is an



- open-source
- distributed
- general-purpose
- cluster-computing

framework designed for 3 purposes:

1. Be easy to use - you can develop applications locally, using a high-level API letting you focus on the content of your computation.
2. Be fast - enabling interactive use and complex algorithms.
3. Be general - allowing to combine multiple types of computations, including text, SQL, graph and machine learning processing (both offline and online) that might previously have required different engines.

All in all, Spark enables to process large quantities of data, beyond what can fit on a single machine, with a high-level, transparent parallelizable, system agnostic and relatively easy-to-use API.

# Introduction to Apache Spark

- Spark started in 2009 as a research project in the UC Berkeley RAD Lab, later to become the AMPLab: <https://amplab.cs.berkeley.edu/>



# Introduction to Apache Spark

- Spark started in 2009 as a research project in the UC Berkeley RAD Lab, later to become the AMPLab: <https://amplab.cs.berkeley.edu/>
- Spark was later donated to the Apache Software Foundation, which has maintained it since: <http://spark.apache.org/>  
Nowadays, Spark has over 1,000 contributors.



# Introduction to Apache Spark

- Spark started in 2009 as a research project in the UC Berkeley RAD Lab, later to become the AMPLab: <https://amplab.cs.berkeley.edu/>
- Spark was later donated to the Apache Software Foundation, which has maintained it since: <http://spark.apache.org/>  
Nowadays, Spark has over 1,000 contributors.
- On top of the Apache avenue, the creators of Spark also created Databricks, a spin-off company leveraging Spark as a cloud-based big data processing tool <https://databricks.com/>. It would be our access point to Spark this semester.



# Introduction to Apache Spark



- Spark is the state-of-the-art computational engine for big data analytics, and it is used by some of the most important companies, including:



- The full list of companies creating products and projects for use with Apache Spark can be seen at: <http://spark.apache.org/powerd-by.html>

# Introduction to Apache Spark



- Spark itself is written in Scala, and runs on the Java Virtual Machine (JVM). It offers simple APIs in the proper Scala, and also in Java, Python and R.



# Introduction to Apache Spark



- Spark itself is written in Scala, and runs on the Java Virtual Machine (JVM). It offers simple APIs in proper Scala, and also in Java, Python and R.
- In this module we are going to work with Spark in Python. Full stop. That being said, for completeness and, in some cases, to ease some explanations, for each code example presented in this module, I will provide you with the Python and Scala versions of it. But, once again, I don't expect you to write a single line of code in Scala.



# Introduction to Apache Spark



- Spark itself is written in Scala, and runs on the Java Virtual Machine (JVM). It offers simple APIs in proper Scala, and also in Java, Python and R.
- In this module we are going to work with Spark in Python. Full stop. That being said, for completeness and, in some cases, to ease some explanations, for each code example presented in this module, I will provide you with the Python and Scala versions of it. But, once again, I don't expect you to write a single line of code in Scala.
- It is important to remark, though, that the Spark code written in Python is often slower than equivalent code written in the JVM, since Scala is statically typed, and the cost of JVM communication (from Python to Scala) can be very high.



# Outline

1. Introduction to Apache Spark.
2. A Spark Application: General Overview.
3. Two Different Roles: IT Manager vs. Data Engineer.
4. Databricks: An Online Platform for Data Engineers.

# A Spark Application: General Overview



*A bit more technical details... Spark in a nutshell*

We said Apache Spark is an open-source, distributed, general-purpose cluster-computing framework designed to be easy to use, fast and general.

To do so, Spark puts together the main concepts we have studied from Distributed programming and some more we will see in the next lecture on functional programming.

# A Spark Application: General Overview

1. Spark extends the notion of computation in a single computer to a cluster of  $n > 1$  computers, which can work together towards a common goal (e.g., our lab in room C127, with its 20 computers working together as a cluster).

# A Spark Application: General Overview

2. Spark extends the notion of algorithm to the one of meta-algorithm / planner, tackling a concrete problem by coordinating other algorithms on the divide, map and reduce stages of solving it.

# A Spark Application: General Overview

2. Spark extends the notion of algorithm to the one of meta-algorithm / planner, tackling a concrete problem by coordinating other algorithms on the divide, map and reduce stages of solving it.
  - Coming back to our lab C127, one of the 20 machines (e.g., machine 1) takes the role of the meta-algorithm (in Spark argot, this means running the driver process).

# A Spark Application: General Overview

2. Spark extends the notion of algorithm to the one of meta-algorithm / planner, tackling a concrete problem by coordinating other algorithms on the divide, map and reduce stages of solving it.
  - Coming back to our lab C127, one of the 20 machines (e.g., machine 1) takes the role of the meta-algorithm (in Spark argot, this means running the driver process).
  - This machine (specifically this driver process) coordinates the remaining 19 machines (e.g., machines 2 - 20) by scheduling, distributing, and monitoring the work of their algorithms (in Spark argot, this means coordinating the 19 executor processes operating on them).

# A Spark Application: General Overview

3. Spark makes the entire distributed computing environment transparent to the user by offering a novel abstract data type: Resilient Distributed Datasets (RDDs).

# A Spark Application: General Overview

3. Spark makes the entire distributed computing environment transparent to the user by offering a novel abstract data type: Resilient Distributed Datasets (RDDs).
  - Thanks to RDDs, a user can write a Spark program as if it was going to be executed in sequential mode by a single machine.

# A Spark Application: General Overview

3. Spark makes the entire distributed computing environment transparent to the user by offering a novel abstract data type: Resilient Distributed Datasets (RDDs).
  - Thanks to RDDs, a user can write a Spark program as if it was going to be executed in sequential mode by a single machine.
  - Then, under the hood, and also thanks to RDDs, Spark scales and distributes the program execution among the different nodes of the cluster.

# A Spark Application: General Overview

3. Spark makes the entire distributed computing environment transparent to the user by offering a novel abstract data type: RDDs.

# A Spark Application: General Overview

3. Spark makes the entire distributed computing environment transparent to the user by offering a novel abstract data type: RDDs.

An RDD is basically a collection of items distributed across many compute nodes that can be manipulated in parallel.

# A Spark Application: General Overview

3. Spark makes the entire distributed computing environment transparent to the user by offering a novel abstract data type: RDDs.

An RDD is basically a collection of items distributed across many compute nodes that can be manipulated in parallel.

- The ADT private part of an RDD is implemented on data structures which are indeed distributed among different nodes of a cluster.

# A Spark Application: General Overview

3. Spark makes the entire distributed computing environment transparent to the user by offering a novel abstract data type: RDDs.

An RDD is basically a collection of items distributed across many compute nodes that can be manipulated in parallel.

- The ADT private part of an RDD is implemented on data structures which are indeed distributed among different nodes of a cluster.
- The complete picture of all existing RDDs, how they are distributed, and the dependencies among them is kept by the driver process.

# A Spark Application: General Overview

3. Spark makes the entire distributed computing environment transparent to the user by offering a novel abstract data type: RDDs.

An RDD is basically a collection of items distributed across many compute nodes that can be manipulated in parallel.

- The ADT private part of an RDD is implemented on data structures which are indeed distributed among different nodes of a cluster.
- The complete picture of all existing RDDs, how they are distributed, and the dependencies among them is kept by the driver process.
- Moreover, these RDDs are tried to be kept in memory as long as possible for their fast access.

# A Spark Application: General Overview

3. Spark makes the entire distributed computing environment transparent to the user by offering a novel abstract data type: RDDs.

An RDD is basically a collection of items distributed across many compute nodes that can be manipulated in parallel.

- However, the ADT public part of an RDD presents it as a unique and indivisible logical unit, offering an extensive API for its access and manipulation.

# A Spark Application: General Overview

3. Spark makes the entire distributed computing environment transparent to the user by offering a novel abstract data type: RDDs.

An RDD is basically a collection of items distributed across many compute nodes that can be manipulated in parallel.

- However, the ADT public part of an RDD presents it as a unique and indivisible logical unit, offering an extensive API for its access and manipulation.

Such this RDD API supports some of the most important functional programming (FP) features: polymorphism, higher-order functions and partial application.

# A Spark Application: General Overview

4. On planning the optimal execution of a program, Spark makes use of the last functional programming feature: lazy evaluation.

# A Spark Application: General Overview

4. On planning the optimal execution of a program, Spark makes use of the last functional programming feature: lazy evaluation.

This lazy evaluation-based computation model turns Spark into a declarative programming framework (even if programs are exposed -written- in imperative programming languages as Python, Java or Scala).

# A Spark Application: General Overview

4. On planning the optimal execution of a program, Spark makes use of the last functional programming feature: lazy evaluation.

This lazy evaluation-based computation model turns Spark into a declarative programming framework (even if programs are exposed -written- in imperative programming languages as Python, Java or Scala).

This declarative nature of Spark provides an explicit isolation between:

# A Spark Application: General Overview

4. On planning the optimal execution of a program, Spark makes use of the last functional programming feature: lazy evaluation.

This lazy evaluation-based computation model turns Spark into a declarative programming framework (even if programs are exposed -written- in imperative programming languages as Python, Java or Scala).

This declarative nature of Spark provides an explicit isolation between:

- Its declarative semantics: the way a program describes -or declares- the properties a solution might have.

# A Spark Application: General Overview

4. On planning the optimal execution of a program, Spark makes use of the last functional programming feature: lazy evaluation.

This lazy evaluation-based computation model turns Spark into a declarative programming framework (even if programs are exposed -written- in imperative programming languages as Python, Java or Scala).

This declarative nature of Spark provides an explicit isolation between:

- Its declarative semantics: the way a program describes -or declares- the properties a solution might have.
- Its operational semantics: the way a program is analysed to schedule/plan the required computations leading to such desired solution.

# A Spark Application: General Overview

4. On planning the optimal execution of a program, Spark makes use of the last functional programming feature: lazy evaluation.
  - On the declarative semantics side we will study the API of the aforementioned RDDs.

# A Spark Application: General Overview

4. On planning the optimal execution of a program, Spark makes use of the last functional programming feature: lazy evaluation.
  - On the declarative semantics side we will study the API of the aforementioned RDDs.
  - In particular, we will study the two classes of operations that allow accessing and manipulating RDDs (transformations and actions).

# A Spark Application: General Overview

4. On planning the optimal execution of a program, Spark makes use of the last functional programming feature: lazy evaluation.
  - On the declarative semantics side we will study the API of the aforementioned RDDs.
  - In particular, we will study the two classes of operations that allow accessing and manipulating RDDs (transformations and actions).
  - All in all, we will study how each Spark program is structured towards the following life-cycle:
    - a. Create some input RDDs from external data.
    - b. Transform them to define new RDDs using transformations.
    - c. Persist any intermediate RDDs that are to be reused.
    - d. Launch actions to kick off a distributed computation.

# A Spark Application: General Overview

4. On planning the optimal execution of a program, Spark makes use of the last functional programming feature: lazy evaluation.
  - On the operational semantics side we will discuss the anatomy of a Spark application via its jobs, stages and tasks.

# A Spark Application: General Overview

4. On planning the optimal execution of a program, Spark makes use of the last functional programming feature: lazy evaluation.
  - On the operational semantics side we will discuss the anatomy of a Spark application via its jobs, stages and tasks.
  - We will discuss the impact of different modelling, RDD partitions and lazy evaluation in the overall performance of a Spark application.

# A Spark Application: General Overview

4. On planning the optimal execution of a program, Spark makes use of the last functional programming feature: lazy evaluation.
  - For now, it suffices to understand that Spark uses lazy evaluation to analyse the program and create a logical plan of action for solving it, which is based on a Direct Acyclic Graph (DAG) with the dependencies among the computations to be performed.

# A Spark Application: General Overview

4. On planning the optimal execution of a program, Spark makes use of the last functional programming feature: lazy evaluation.
  - For now, it suffices to understand that Spark uses lazy evaluation to analyse the program and create a logical plan of action for solving it, which is based on a Direct Acyclic Graph (DAG) with the dependencies among the computations to be performed.
  - Such DAG is organised towards atomically (indivisible) units of computation to be performed on partitions of RDDs. They are called tasks, and each task is assigned to a single core/executor process (in a single machine).

# A Spark Application: General Overview

4. On planning the optimal execution of a program, Spark makes use of the last functional programming feature: lazy evaluation.
  - For now, it suffices to understand that Spark uses lazy evaluation to analyse the program and create a logical plan of action for solving it, which is based on a Direct Acyclic Graph (DAG) with the dependencies among the computations to be performed.
  - Such DAG is organised towards atomically (indivisible) units of computation to be performed on partitions of RDDs. They are called tasks, and each task is assigned to a single core/executor process (in a single machine).
  - The DAG is further optimised and organised towards stages (groups of tasks that can be pipelined together).

# A Spark Application: General Overview

4. On planning the optimal execution of a program, Spark makes use of the last functional programming feature: lazy evaluation.
  - For now, it suffices to understand that Spark uses lazy evaluation to analyse the program and create a logical plan of action for solving it, which is based on a Direct Acyclic Graph (DAG) with the dependencies among the computations to be performed.
  - Such DAG is organised towards atomically (indivisible) units of computation to be performed on partitions of RDDs. They are called tasks, and each task is assigned to a single core/executor process (in a single machine).
  - The DAG is further optimised and organised towards stages (groups of tasks that can be pipelined together).
  - Finally, the logical plan stated by the DAG is translated into a physical plan by scheduling and tracking the execution of the tasks over the computers of the cluster.

# Outline

1. Introduction to Apache Spark.
2. A Spark Application: General Overview.
3. Two Different Roles: IT Manager vs. Data Engineer.
4. Databricks: An Online Platform for Data Engineers.

# Two Different Roles: IT Manager vs. Data Engineer

- The fact of Spark being a cluster-computing framework for data analysis requires to have a cluster on the first place.

## Two Different Roles: IT Manager vs. Data Engineer

- The fact of Spark being a cluster-computing framework for data analysis requires to have a cluster on the first place.
- Having a cluster of  $n > 1$  computers is not complicated, but to have it "*ready*" for using Spark has additional requirements:

## Two Different Roles: IT Manager vs. Data Engineer

- The fact of Spark being a cluster-computing framework for data analysis requires to have a cluster on the first place.
- Having a cluster of  $n > 1$  computers is not complicated, but to have it "*ready*" for using Spark has additional requirements:
  1. All computers must be able to communicate.

# Two Different Roles: IT Manager vs. Data Engineer

- The fact of Spark being a cluster-computing framework for data analysis requires to have a cluster on the first place.
- Having a cluster of  $n > 1$  computers is not complicated, but to have it "*ready*" for using Spark has additional requirements:
  1. All computers must be able to communicate.
  2. A common Spark version has to be installed across the cluster.

# Two Different Roles: IT Manager vs. Data Engineer

- The fact of Spark being a cluster-computing framework for data analysis requires to have a cluster on the first place.
- Having a cluster of  $n > 1$  computers is not complicated, but to have it "*ready*" for using Spark has additional requirements:
  1. All computers must be able to communicate.
  2. A common Spark version has to be installed across the cluster.
  3. A Distributed File System has to be in place to offer a global storage view of the cluster.

# Two Different Roles: IT Manager vs. Data Engineer

- The fact of Spark being a cluster-computing framework for data analysis requires to have a cluster on the first place.
- Having a cluster of  $n > 1$  computers is not complicated, but to have it "*ready*" for using Spark has additional requirements:
  1. All computers must be able to communicate.
  2. A common Spark version has to be installed across the cluster.
  3. A Distributed File System has to be in place to offer a global storage view of the cluster.
  4. OS and security updates have to be taken into account as well.

# Two Different Roles: IT Manager vs. Data Engineer

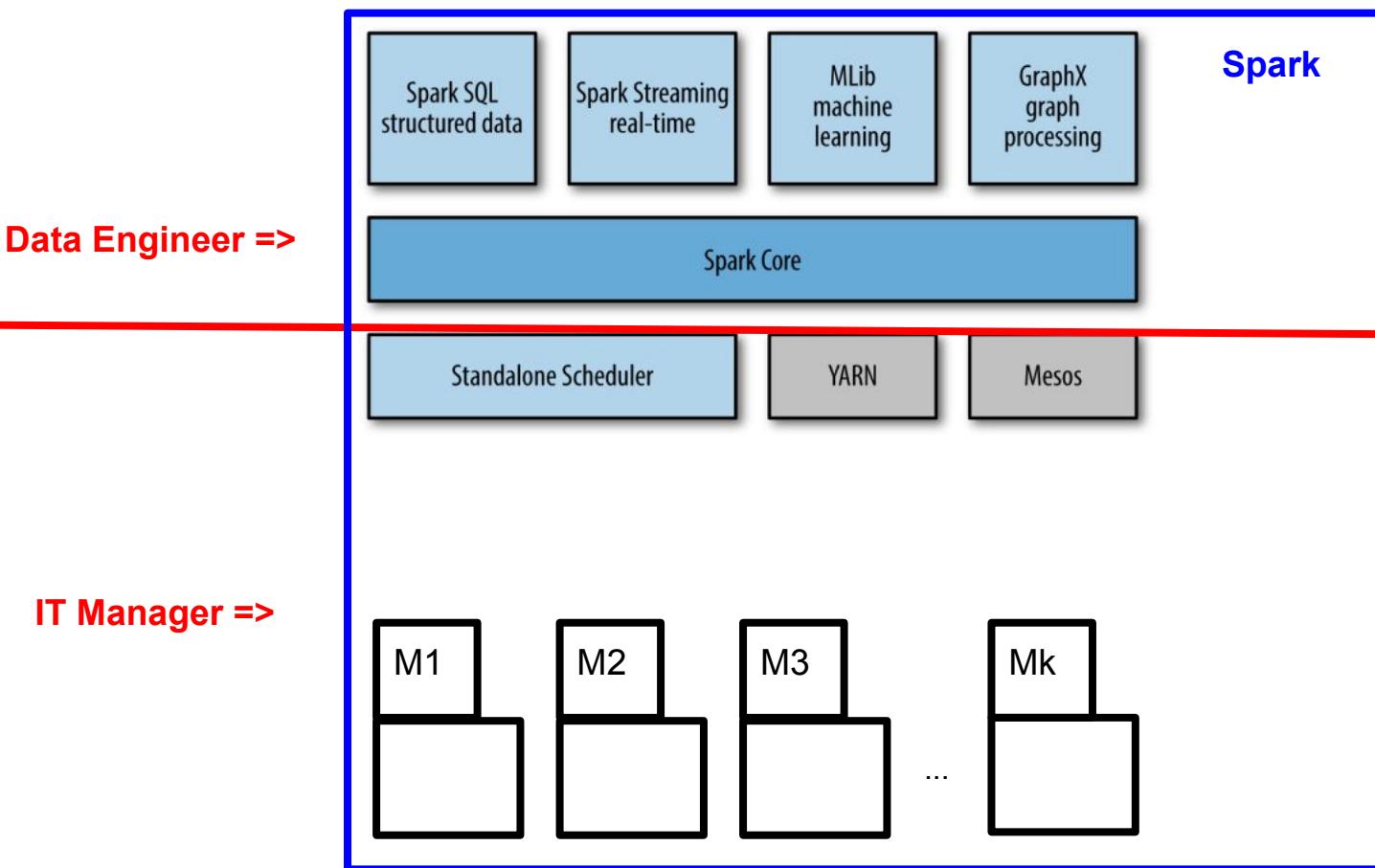
- The fact of Spark being a cluster-computing framework for data analysis requires to have a cluster on the first place.
- Having a cluster of  $n > 1$  computers is not complicated, but to have it "*ready*" for using Spark has additional requirements:
  1. All computers must be able to communicate.
  2. A common Spark version has to be installed across the cluster.
  3. A Distributed File System has to be in place to offer a global storage view of the cluster.
  4. OS and security updates have to be taken into account as well.
- All these tasks make room for a role as Cluster IT Manager, which is out of the scope of this module, and thus we are not going to cover at all.

# Two Different Roles: IT Manager vs. Data Engineer

- The fact of Spark being a cluster-computing framework for data analysis requires to have a cluster on the first place.
- Having a cluster of  $n > 1$  computers is not complicated, but to have it "*ready*" for using Spark has additional requirements:
  1. All computers must be able to communicate.
  2. A common Spark version has to be installed across the cluster.
  3. A Distributed File System has to be in place to offer a global storage view of the cluster.
  4. OS and security updates have to be taken into account as well.
- All these tasks make room for a role as Cluster IT Manager, which is out of the scope of this module, and thus we are not going to cover at all.  
If you want to learn more about the topic perhaps a good book is:
  - ❖ Spark Operations: Operationalizing Apache Spark at Scale.  
*Timothy Chen et al. O'Reilly, 2016.*

# Two Different Roles: IT Manager vs. Data Engineer

- We are going to be 100% focused on the role of the Data Scientist or Data Engineer, who uses Spark on top of a "ready" cluster.



# Two Different Roles: IT Manager vs. Data Engineer

- As we saw in the previous picture, Spark contains multiple components, some of them IT Manager-related and some Data Engineer-related.

# Two Different Roles: IT Manager vs. Data Engineer

- As we saw in the previous picture, Spark contains multiple components, some of them IT Manager-related and some Data Engineer-related.
- On the Data Engineer side of things, the amount of components aims to efficiently support a wide range of workloads, including stream processing, graph-based processing, machine learning processing (and any combination of the aforementioned).

# Two Different Roles: IT Manager vs. Data Engineer

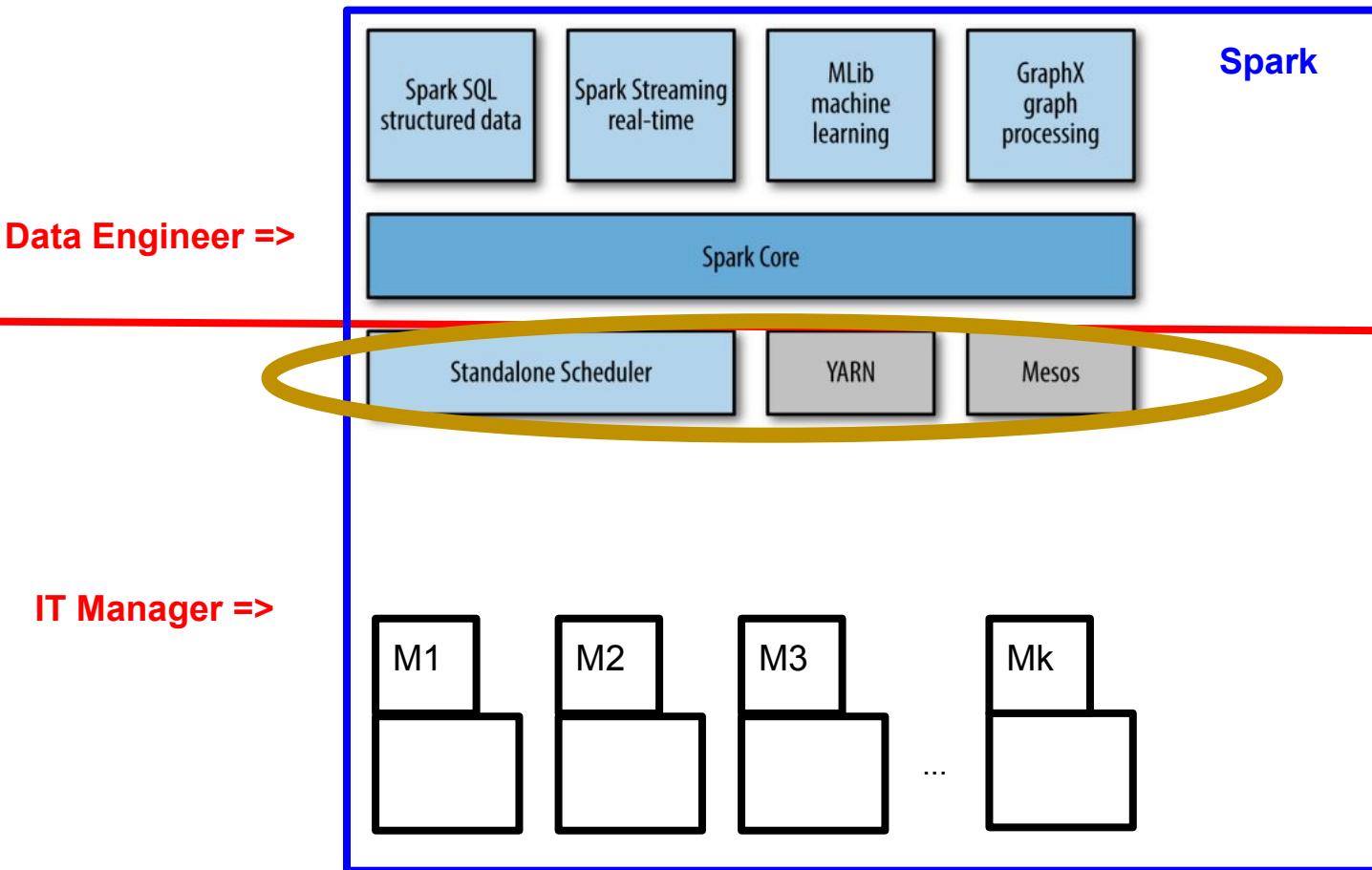
- As we saw in the previous picture, Spark contains multiple components, some of them IT Manager-related and some Data Engineer-related.
- On the Data Engineer side of things, the amount of components aims to efficiently support a wide range of workloads, including stream processing, graph-based processing, machine learning processing (and any combination of the aforementioned).
- By supporting these workloads in the same engine, Spark makes it easy and inexpensive to combine different processing types, which is handy for domain applications requiring the use of several of them.

# Two Different Roles: IT Manager vs. Data Engineer

- As we saw in the previous picture, Spark contains multiple components, some of them IT Manager-related and some Data Engineer-related.
- On the Data Engineer side of things, the amount of components aims to efficiently support a wide range of workloads, including stream processing, graph-based processing, machine learning processing (and any combination of the aforementioned).
- By supporting these workloads in the same engine, Spark makes it easy and inexpensive to combine different processing types, which is handy for domain applications requiring the use of several of them.
- We describe each of these components now...

# Two Different Roles: IT Manager vs. Data Engineer

- IT Manager => Cluster Manager Component



## Two Different Roles: IT Manager vs. Data Engineer

- On its own, Spark is not a data storage solution; it performs computations on Java Virtual Machines (JVMs) that last only for the duration of a Spark application.

## Two Different Roles: IT Manager vs. Data Engineer

- On its own, Spark is not a data storage solution; it performs computations on Java Virtual Machines (JVMs) that last only for the duration of a Spark application.
- Spark can be run locally on a single machine with a single JVM (called local mode). However, Spark is designed to efficiently scale up from one to many thousands of compute nodes.

## Two Different Roles: IT Manager vs. Data Engineer

- On its own, Spark is not a data storage solution; it performs computations on Java Virtual Machines (JVMs) that last only for the duration of a Spark application.
- Spark can be run locally on a single machine with a single JVM (called local mode). However, Spark is designed to efficiently scale up from one to many thousands of compute nodes.
- When used in a cluster, Spark is used in tandem with a distributed storage system and a cluster manager.

# Two Different Roles: IT Manager vs. Data Engineer

- On its own, Spark is not a data storage solution; it performs computations on Java Virtual Machines (JVMs) that last only for the duration of a Spark application.
- Spark can be run locally on a single machine with a single JVM (called local mode). However, Spark is designed to efficiently scale up from one to many thousands of compute nodes.
- When used in a cluster, Spark is used in tandem with a distributed storage system and a cluster manager.
  - The storage system is used to provide the input dataset used by Spark application, as well as to stable store the results such application produces.

# Two Different Roles: IT Manager vs. Data Engineer

- On its own, Spark is not a data storage solution; it performs computations on Java Virtual Machines (JVMs) that last only for the duration of a Spark application.
- Spark can be run locally on a single machine with a single JVM (called local mode). However, Spark is designed to efficiently scale up from one to many thousands of compute nodes.
- When used in a cluster, Spark is used in tandem with a distributed storage system and a cluster manager.
  - The storage system is used to provide the input dataset used by Spark application, as well as to stable store the results such application produces.
  - The cluster manager is used to orchestrate the distribution of Spark applications across the cluster.

## Two Different Roles: IT Manager vs. Data Engineer

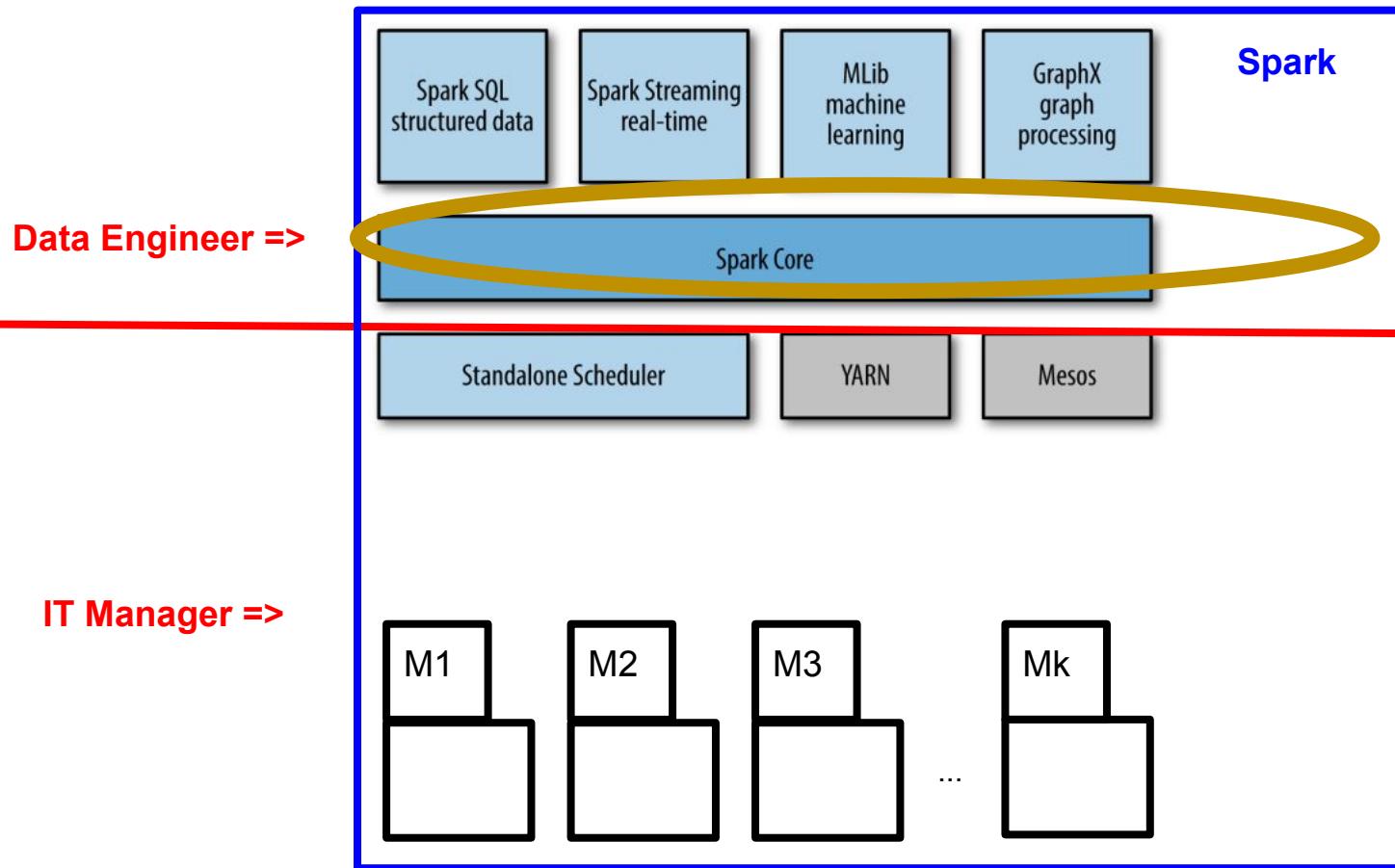
- This cluster manager is in charge of putting in contact the storage of the dataset with its further processing (via one or more of the data engineering components).  
Spark can run over a variety of cluster managers, including Hadoop YARN, Apache Mesos, and a simple cluster manager included in Spark itself called the Standalone Scheduler.

## Two Different Roles: IT Manager vs. Data Engineer

- This cluster manager is in charge of putting in contact the storage of the dataset with its further processing (via one or more of the data engineering components).  
Spark can run over a variety of cluster managers, including Hadoop YARN, Apache Mesos, and a simple cluster manager included in Spark itself called the Standalone Scheduler.
- In terms of file storage, Spark can work on Hadoop Distributed File System (HDFS) or similar (when using Spark with the Databricks online system we will use their specific Databricks File System - DBFS).

# Two Different Roles: IT Manager vs. Data Engineer

- Data Engineer => Spark Core



# Two Different Roles: IT Manager vs. Data Engineer

- Spark Core contains the basic functionality of Spark, including:

# Two Different Roles: IT Manager vs. Data Engineer

- Spark Core contains the basic functionality of Spark, including:
  - The main data abstraction being offered to users to express their programs: The Resilient Distributed Datasets (RDDs).

# Two Different Roles: IT Manager vs. Data Engineer

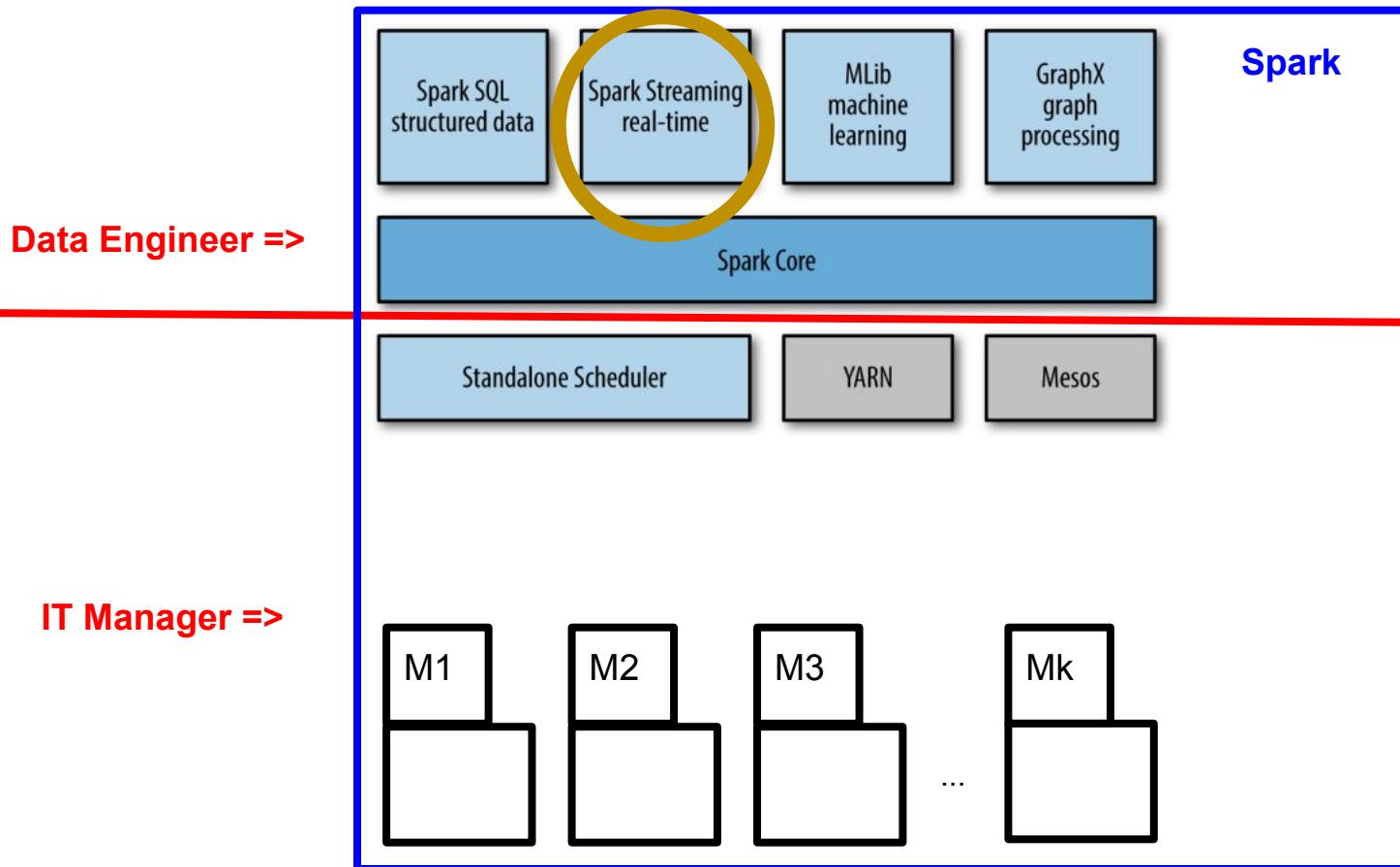
- Spark Core contains the basic functionality of Spark, including:
  - The main data abstraction being offered to users to express their programs: The Resilient Distributed Datasets (RDDs).
  - The implementation of the driver and executor processes that Spark uses to run each application.
    - This includes the Spark driver functionality to generate a logical plan (Direct Acyclic Graph per job, with concrete stages and tasks) and a physical plan (schedule and tracking of the tasks execution).

# Two Different Roles: IT Manager vs. Data Engineer

- Spark Core contains the basic functionality of Spark, including:
  - The main data abstraction being offered to users to express their programs: The Resilient Distributed Datasets (RDDs).
  - The implementation of the driver and executor processes that Spark uses to run each application.
    - This includes the Spark driver functionality to generate a logical plan (Direct Acyclic Graph per job, with concrete stages and tasks) and a physical plan (schedule and tracking of the tasks execution).
  - Other functionality such as memory management, fault recovery and the interaction with storage systems.

# Two Different Roles: IT Manager vs. Data Engineer

- Data Engineer => Spark Streaming



# Two Different Roles: IT Manager vs. Data Engineer

- Spark Streaming is a Spark component that enables processing of live streams of data.
  - Examples of data streams include log-files generated by production web servers, or queues of messages containing status updates posted by users of a web service.

# Two Different Roles: IT Manager vs. Data Engineer

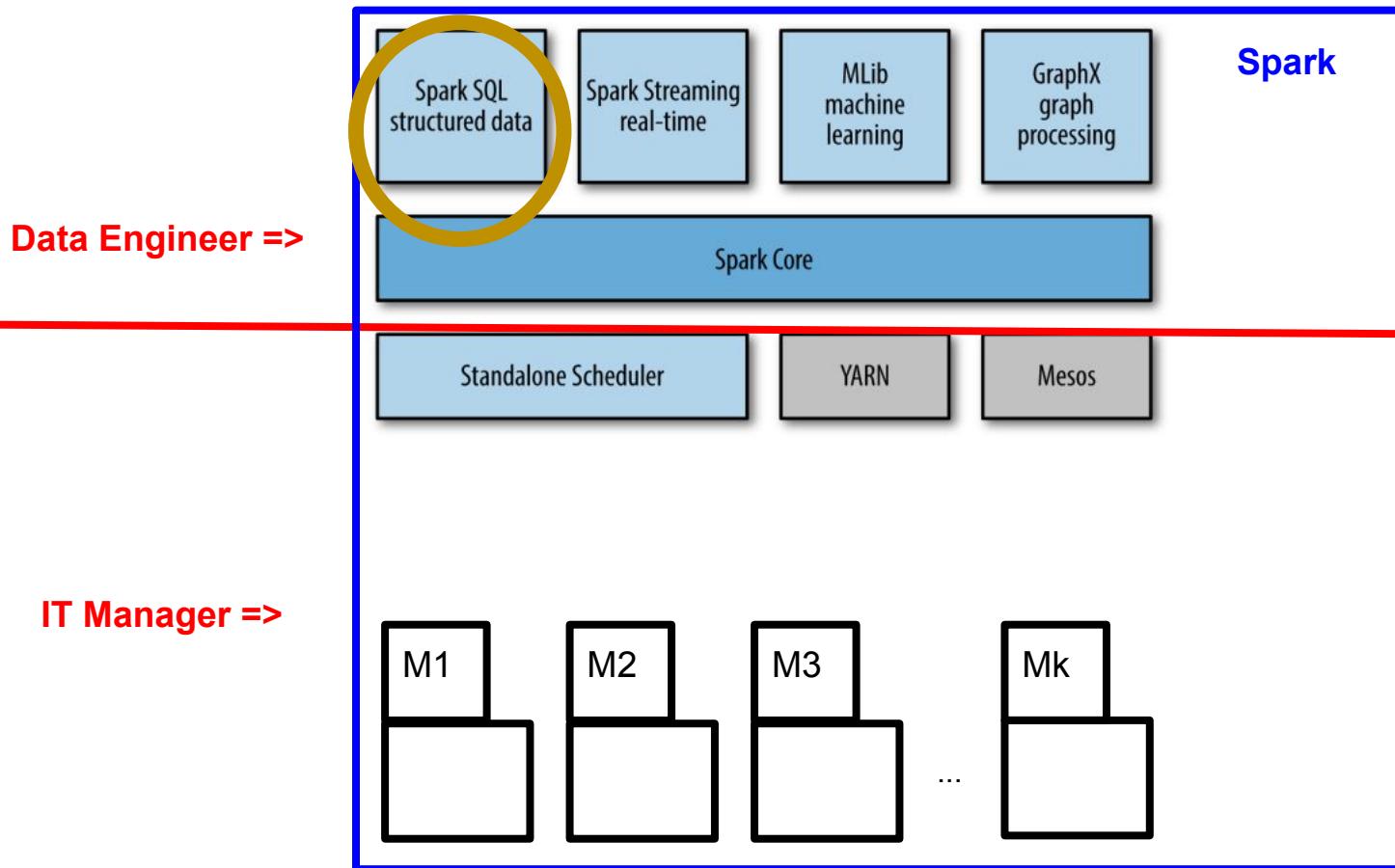
- Spark Streaming is a Spark component that enables processing of live streams of data.
  - Examples of data streams include log-files generated by production web servers, or queues of messages containing status updates posted by users of a web service.
- The main data abstraction being offered is called DStream (from discretised stream), and is based on the amalgamation of RDDs over time via batch intakes.

# Two Different Roles: IT Manager vs. Data Engineer

- Spark Streaming is a Spark component that enables processing of live streams of data.
  - Examples of data streams include log-files generated by production web servers, or queues of messages containing status updates posted by users of a web service.
- The main data abstraction being offered is called DStream (from discretised stream), and is based on the amalgamation of RDDs over time via batch intakes.
- The library offers a good range of stateless and stateful operations. However, as is the case with all RDD-based components, it has not been developed further to entice users to use its counterpart DataFrame-based components.

# Two Different Roles: IT Manager vs. Data Engineer

- Data Engineer => Spark SQL



# Two Different Roles: IT Manager vs. Data Engineer

- Spark SQL is the module integrating relational processing with Spark's functional programming API.

# Two Different Roles: IT Manager vs. Data Engineer

- Spark SQL is the module integrating relational processing with Spark's functional programming API.
- The users benefit of a higher-level data abstraction (DataFrames) for ingesting, querying and persisting (semi)structured data using relational queries via a Domain Specific Language (DSL).

# Two Different Roles: IT Manager vs. Data Engineer

- Spark SQL is the module integrating relational processing with Spark's functional programming API.
- The users benefit of a higher-level data abstraction (DataFrames) for ingesting, querying and persisting (semi)structured data using relational queries via a Domain Specific Language (DSL).
- Under the hood, Spark SQL translates a DataFrame-based program into an equivalent RDD-based one via:

# Two Different Roles: IT Manager vs. Data Engineer

- Spark SQL is the module integrating relational processing with Spark's functional programming API.
- The users benefit of a higher-level data abstraction (DataFrames) for ingesting, querying and persisting (semi)structured data using relational queries via a Domain Specific Language (DSL).
- Under the hood, Spark SQL translates a DataFrame-based program into an equivalent RDD-based one via:
  - A Catalyst optimiser (which semantically analyse the query expressed by the user).

# Two Different Roles: IT Manager vs. Data Engineer

- Spark SQL is the module integrating relational processing with Spark's functional programming API.
- The users benefit of a higher-level data abstraction (DataFrames) for ingesting, querying and persisting (semi)structured data using relational queries via a Domain Specific Language (DSL).
- Under the hood, Spark SQL translates a DataFrame-based program into an equivalent RDD-based one via:
  - A Catalyst optimiser (which semantically analyse the query expressed by the user).
  - A Tungsten encoder optimiser (which translates the query to an equivalent binary RDD-based format).

## Two Different Roles: IT Manager vs. Data Engineer

- As mentioned before, Spark SQL has also incorporated its own library for processing streaming datasets, called Structured Streaming.

## Two Different Roles: IT Manager vs. Data Engineer

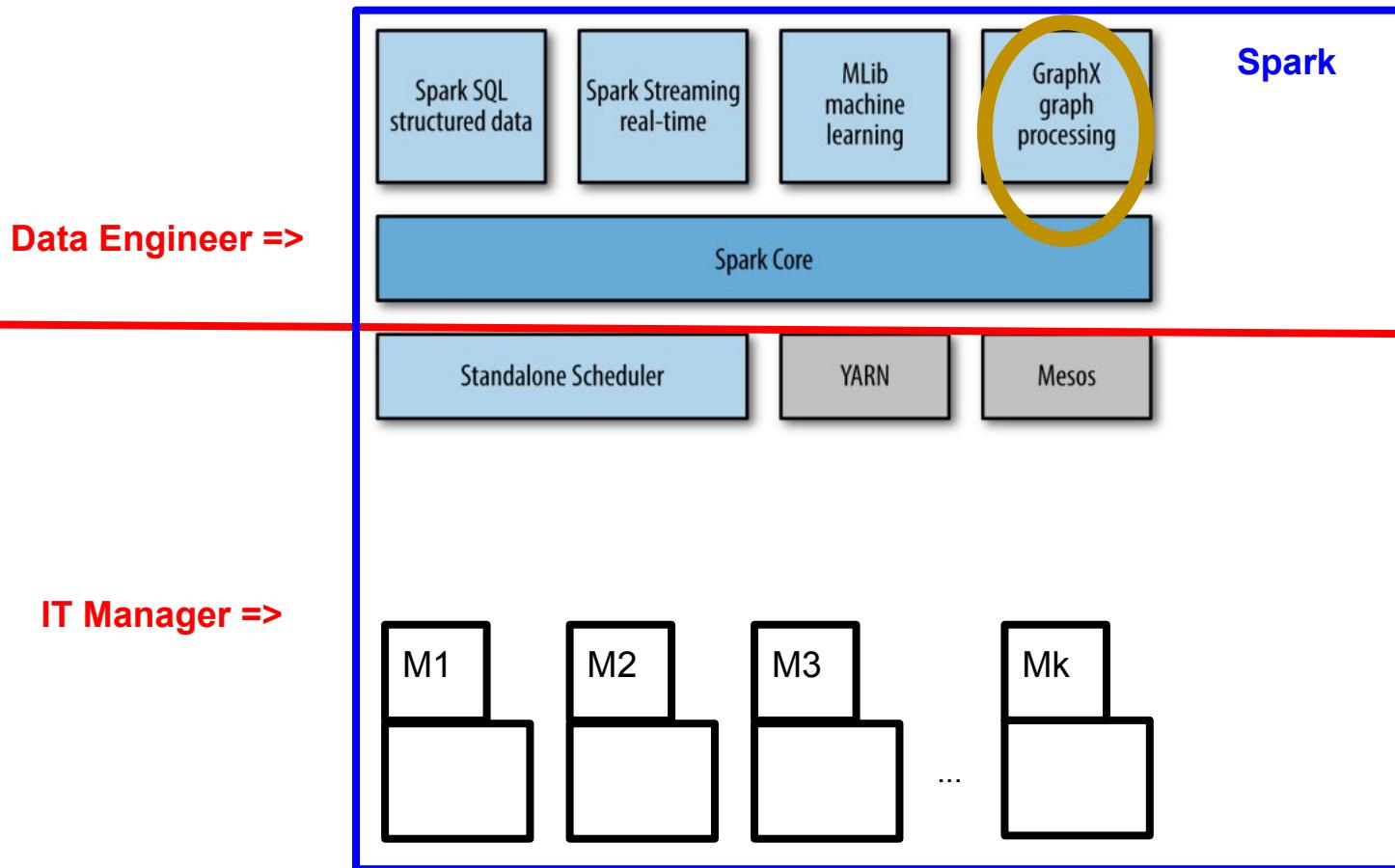
- As mentioned before, Spark SQL has also incorporated its own library for processing streaming datasets, called Structured Streaming.
- Based on DataFrames, it allows users to express a streaming computation in a transparent way, just as a batch computation on static data. The Spark SQL engine takes care of running it incrementally and continuously and updating the final result as streaming data continues to arrive.

## Two Different Roles: IT Manager vs. Data Engineer

- As mentioned before, Spark SQL has also incorporated its own library for processing streaming datasets, called Structured Streaming.
- Based on DataFrames, it allows users to express a streaming computation in a transparent way, just as a batch computation on static data. The Spark SQL engine takes care of running it incrementally and continuously and updating the final result as streaming data continues to arrive.
- The system also supports managing late-time data arrival offering end-to-end exactly-once fault-tolerance guarantees through checkpointing and write ahead Logs.

# Two Different Roles: IT Manager vs. Data Engineer

- Data Engineer => Graph



# Two Different Roles: IT Manager vs. Data Engineer

- GraphX is a library for manipulating graphs (e.g., a social network's friend graph) and performing graph-parallel computations.

# Two Different Roles: IT Manager vs. Data Engineer

- GraphX is a library for manipulating graphs (e.g., a social network's friend graph) and performing graph-parallel computations.
- GraphX extends the Spark RDD API (allowing us to create directed graphs with arbitrary properties attached to each vertex and edge) and provides a library with common graph algorithms.
-

## Two Different Roles: IT Manager vs. Data Engineer

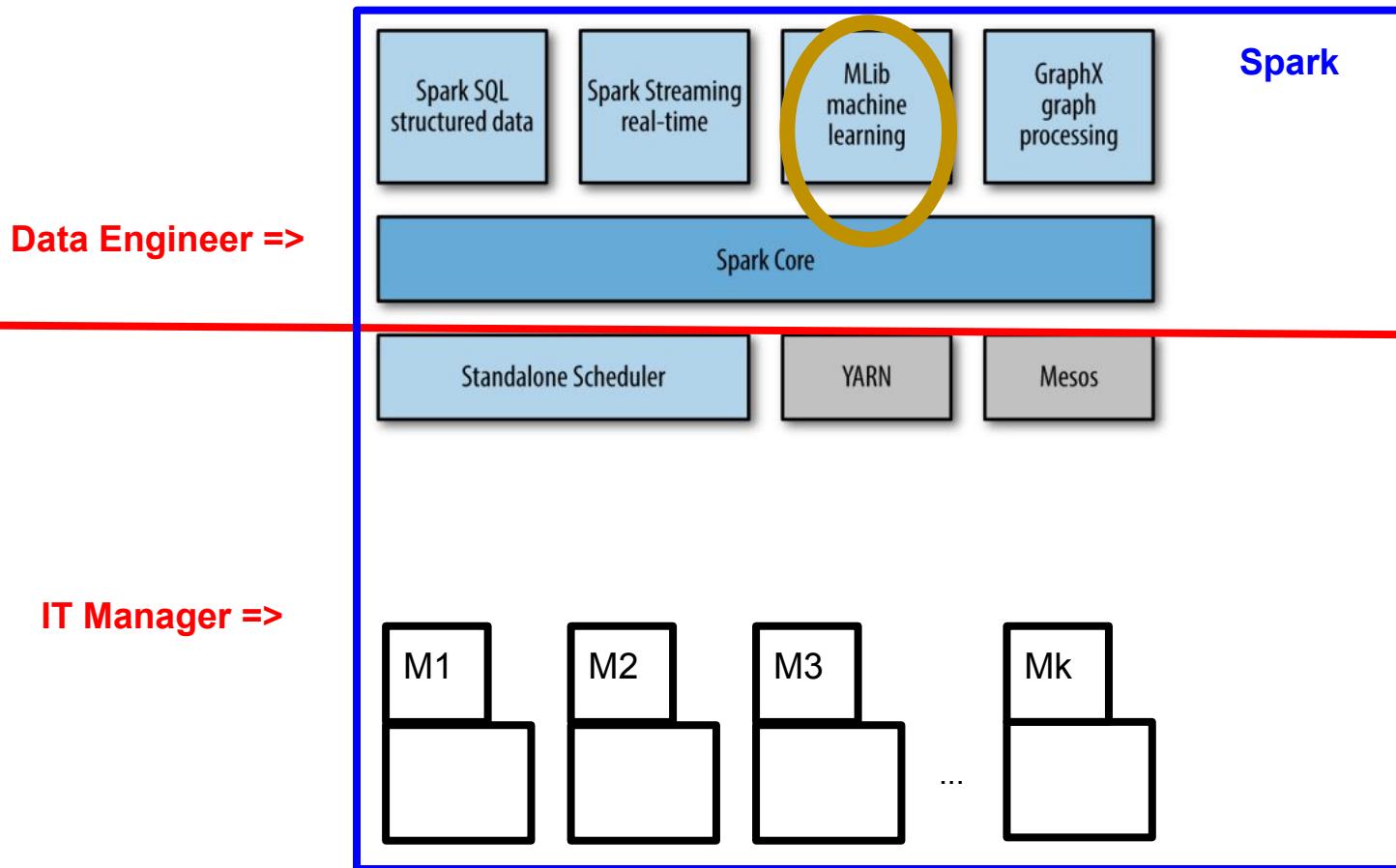
- GraphX is a library for manipulating graphs (e.g., a social network's friend graph) and performing graph-parallel computations.
- GraphX extends the Spark RDD API (allowing us to create directed graphs with arbitrary properties attached to each vertex and edge) and provides a library with common graph algorithms.
- Unfortunately this library is only available via Scala and Java, and thus for working in Python we will use its DataFrame counterpart GraphFrames. As its name indicates, this library is based on the DataFrames abstraction, although it provides its own graph-based language notation for users to expressing their programs.

## Two Different Roles: IT Manager vs. Data Engineer

- GraphX is a library for manipulating graphs (e.g., a social network's friend graph) and performing graph-parallel computations.
- GraphX extends the Spark RDD API (allowing us to create directed graphs with arbitrary properties attached to each vertex and edge) and provides a library with common graph algorithms.
- Unfortunately this library is only available via Scala and Java, and thus for working in Python we will use its DataFrame counterpart GraphFrames. As its name indicates, this library is based on the DataFrames abstraction, although it provides its own graph-based language notation for users to expressing their programs.
- It is very unlikely that we will have time to cover this component at all in our 22 lectures. However, I will pass you on all the code examples I have created for it.

# Two Different Roles: IT Manager vs. Data Engineer

- Data Engineer => MLlib



## Two Different Roles: IT Manager vs. Data Engineer

- Finally, Spark also comes with components for common machine learning functionality, called MLLib (RDD-based) and ML (DataFrame-based).

## Two Different Roles: IT Manager vs. Data Engineer

- Finally, Spark also comes with components for common machine learning functionality, called MLLib (RDD-based) and ML (DataFrame-based).
- They provide multiple types of machine learning algorithms, including classification, regression, clustering, and collaborative filtering, as well as supporting functionality such as model evaluation and data import. All of these methods are designed to scale out across a cluster.

## Two Different Roles: IT Manager vs. Data Engineer

- Finally, Spark also comes with components for common machine learning functionality, called MLLib (RDD-based) and ML (DataFrame-based).
- They provide multiple types of machine learning algorithms, including classification, regression, clustering, and collaborative filtering, as well as supporting functionality such as model evaluation and data import. All of these methods are designed to scale out across a cluster.
- It is very unlikely that we will have time to cover this component at all in our 22 lectures. Indeed, I am only starting learning it myself. However, if I have time to create some code examples for it I will pass them on to you at the end of the semester.

# Outline

1. Introduction to Apache Spark.
2. A Spark Application: General Overview.
3. Two Different Roles: IT Manager vs. Data Engineer.
4. **Databricks: An Online Platform for Data Engineers.**

# Databricks: An Online Platform for Data Engineers

- Databricks is a spin-off company leveraging Spark as a **cloud-based** big data processing tool <https://databricks.com/>



databricks<sup>®</sup>

# Databricks: An Online Platform for Data Engineers

- Databricks is a spin-off company leveraging Spark as a **cloud-based** big data processing tool <https://databricks.com/>



databricks<sup>®</sup>

- Databricks abstracts the IT management tasks by allowing to set up and configure a cluster with just a few clicks.

# Databricks: An Online Platform for Data Engineers

- Databricks is a spin-off company leveraging Spark as a **cloud-based** big data processing tool <https://databricks.com/>



databricks<sup>®</sup>

- Databricks abstracts the IT management tasks by allowing to set up and configure a cluster with just a few clicks.
- It provides an easy interface for performing Data Engineering tasks (import, edit and running Spark applications) on top of the cluster being created.

# Databricks: An Online Platform for Data Engineers

- The Databricks Community Edition is free to use.  
It will be our approach in this module.



# Databricks: An Online Platform for Data Engineers

- The Databricks Community Edition is free to use.  
It will be our approach in this module.
- The Community Edition provides us with a local mode-based  
Spark configuration (similar to using Spark in our own local machine).



# Databricks: An Online Platform for Data Engineers

- The Databricks Community Edition is free to use.  
It will be our approach in this module.
- The Community Edition provides us with a local mode-based Spark configuration (similar to using Spark in our own local machine).
- Specifically, it provides us with **A Cluster of 1 Databricks Unit (DBU)**:  
*“A unit of processing capability per hour”*
  - 1 physical machine with
  - 2 cores and
  - 8GB of memory storage.



# Databricks: An Online Platform for Data Engineers

- On the other hand, the Databricks Enterprise Edition allows us to rent a cluster in a price-per-usage basis, and configure it to target the Business Unit specific needs.



# Databricks: An Online Platform for Data Engineers

- On the other hand, the Databricks Enterprise Edition allows us to rent a cluster in a price-per-usage basis, and configure it to target the Business Unit specific needs.
- The cluster is indeed hosted by Microsoft Azure or Amazon Web Services, with Databricks providing the interface to automate the cluster setup.



Microsoft  
Azure



# Databricks: An Online Platform for Data Engineers

- On the other hand, the Databricks Enterprise Edition allows us to rent a cluster in a price-per-usage basis, and configure it to target the Business Unit specific needs.
- The cluster is indeed hosted by Microsoft Azure or Amazon Web Services, with Databricks providing the interface to automate the cluster setup.



Microsoft  
Azure



- The range of prices depends on the functionality provided by Databricks, and can be consulted at: <https://databricks.com/product/pricing#dbu>



# Databricks: An Online Platform for Data Engineers

*How to...*  
Sign up / sign in.

# Databricks: An Online Platform for Data Engineers

1. We register/sign up to Databricks Community Edition at:  
<https://databricks.com/signup/signup-community>

 databricks Platform Solutions Customers Learn Partners Events Open Source Company

## Sign Up for Databricks Community Edition

First Name \*

Last Name \*

Company Name \*

Work Email \*

Phone Number

What is your intended use case? \*

How would you describe your role? \*

Keep me informed with the occasional update about Databricks and Apache Spark™.

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

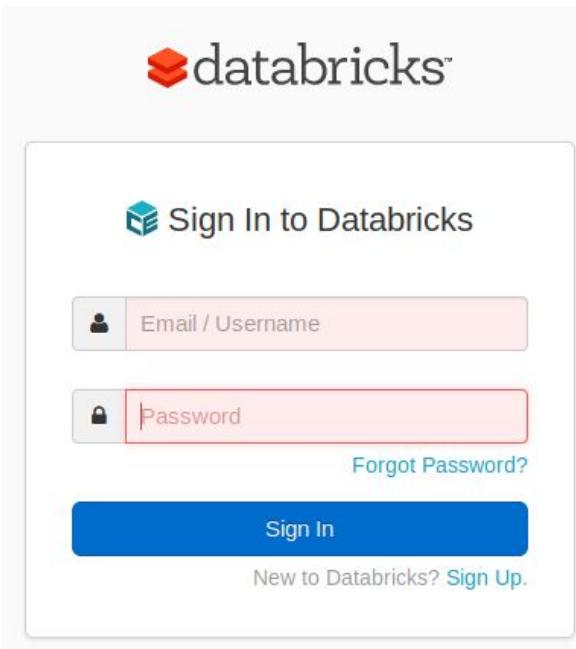
I'm not a robot   
reCAPTCHA  
Privacy • Terms

[Sign Up](#)

# Databricks: An Online Platform for Data Engineers

- Once registered, we sign in at:

<https://community.cloud.databricks.com>



# Databricks: An Online Platform for Data Engineers

*How to...*  
Create a new cluster.

# Databricks: An Online Platform for Data Engineers

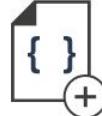
3. We are redirected to the main page of the web interface. Our first step is to set up a new Community Edition cluster by clicking in **Clusters**.



Welcome to databricks™

 [Explore the Quickstart Tutorial](#)  
Spin up a cluster, run queries on preloaded data, and display results in 5 minutes.

 [Import & Explore Data](#)  
Quickly import data, preview its schema, create a table, and query it in a notebook.

 [Create a Blank Notebook](#)  
Create a notebook to start querying, visualizing, and modeling your data.

**Common Tasks**

-  [New Notebook](#)
-  [Create Table](#)
-  [New Cluster](#)

**Recents**

-  [34\\_job\\_inspection\\_5.scala](#)
-  [33\\_job\\_inspection\\_4.scala](#)
-  [32\\_job\\_inspection\\_3.scala](#)

**What's new in v3.2**

- Instance Pools
- Databricks Runtime 6.0 will drop Python 2 support

[View latest release notes](#)

# Databricks: An Online Platform for Data Engineers

3. We set up the new cluster with the following parameters:

The screenshot shows the 'Create Cluster' interface in Databricks. On the left is a sidebar with icons for Home, Workspace, Recents, Data, Clusters, Jobs, and Search. The main area has a title 'Create Cluster' and a sub-section 'New Cluster'. It includes a 'Cancel' button and a 'Create Cluster' button (which is highlighted in blue). Below these are sections for 'Cluster Name' (containing 'MyCluster'), 'Databricks Runtime Version' (set to 'Runtime: 5.5 LTS (Scala 2.11, Spark 2.4.3)'), 'Python Version' (set to '3'), and 'Instance' (with a note about free memory and a link to upgrade the subscription). At the bottom, there are tabs for 'Instances' (selected) and 'Spark', and a dropdown for 'Availability Zone' set to 'us-west-2c'.

Create Cluster

New Cluster | Cancel | Create Cluster

0 Workers: 0.0 GB Memory, 0 Cores, 0 DBU  
1 Driver: 6.0 GB Memory, 0.88 Cores, 1 DBU

Cluster Name

MyCluster

Databricks Runtime Version

Runtime: 5.5 LTS (Scala 2.11, Spark 2.4.3)

Python Version

3

Instance

Free 6GB 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

# Databricks: An Online Platform for Data Engineers

3. We set up the new cluster with the following parameters:

The screenshot shows the Databricks 'Create Cluster' interface. On the left is a sidebar with icons for Home, Workspace, Recents, Data, Clusters, Jobs, and Search. The main area has a title 'Create Cluster' and a sub-section 'New Cluster'. It includes a 'Cancel' button and a 'Create Cluster' button. To the right of the buttons are resource specifications: '0 Workers: 0.0 GB Memory, 0 Cores, 0 DBU' and '1 Driver: 6.0 GB Memory, 0.88 Cores, 1 DBU'. Below these are several configuration fields:

- Cluster Name:** A text input field containing 'MyCluster', which is highlighted with a red rectangle.
- Databricks Runtime Version:** A dropdown menu showing 'Runtime: 5.5 LTS (Scala 2.11, Spark 2.4.3)'.
- Python Version:** A dropdown menu showing '3'.
- Instance:** A note stating 'Free 6GB Memory: As a Community Edition user, your cluster will automatically terminate after an idle period of two hours.' followed by a link 'For more configuration options, please [upgrade your Databricks subscription](#)'.
- Instances:** A tabbed section showing 'Instances' (selected) and 'Spark'.
- Availability Zone:** A dropdown menu showing 'us-west-2c'.

# Databricks: An Online Platform for Data Engineers

3. We set up the new cluster with the following parameters:

The screenshot shows the 'Create Cluster' interface in Databricks. On the left is a sidebar with icons for Home, Workspace, Recents, Data, Clusters, Jobs, and Search. The main area has a title 'Create Cluster' and a sub-section 'New Cluster'. It includes a 'Cancel' button and a 'Create Cluster' button. Below these are sections for 'Cluster Name' (containing 'MyCluster'), 'Databricks Runtime Version' (set to 'Runtime: 5.5 LTS (Scala 2.11, Spark 2.4.3)', highlighted with a red box), 'Python Version' (set to '3'), 'Instance' (with a note about free memory and upgrade options), and 'Availability Zone' (set to 'us-west-2c').

Create Cluster

New Cluster | Cancel | Create Cluster

0 Workers: 0.0 GB Memory, 0 Cores, 0 DBU  
1 Driver: 6.0 GB Memory, 0.88 Cores, 1 DBU

Cluster Name

MyCluster

Databricks Runtime Version

Runtime: 5.5 LTS (Scala 2.11, Spark 2.4.3)

Python Version

3

Instance

Free 6GB 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

# Databricks: An Online Platform for Data Engineers

3. We set up the new cluster with the following parameters:

The screenshot shows the 'Create Cluster' interface in Databricks. On the left is a sidebar with icons for Home, Workspace, Recents, Data, Clusters, Jobs, and Search. The main area has a title 'New Cluster' with a 'Cancel' button and a 'Create Cluster' button. To the right of the button are cluster specifications: '0 Workers: 0.0 GB Memory, 0 Cores, 0 DBU' and '1 Driver: 6.0 GB Memory, 0.88 Cores, 1 DBU'. Below these are fields for 'Cluster Name' (MyCluster), 'Databricks Runtime Version' (Runtime: 5.5 LTS (Scala 2.11, Spark 2.4.3)), and 'Python Version' (set to 3). A red box highlights the 'Python Version' dropdown. At the bottom, there's an 'Instance' section with a note about free memory and a link to upgrade the subscription, and tabs for 'Instances' (selected) and 'Spark'.

# Databricks: An Online Platform for Data Engineers

3. We set up the new cluster with the following parameters:

The screenshot shows the 'Create Cluster' interface in the Databricks web application. On the left, a sidebar menu lists 'databricks', 'Home', 'Workspace', 'Recents', 'Data', 'Clusters', 'Jobs', and 'Search'. The main area is titled 'Create Cluster' and 'New Cluster'. It includes fields for 'Cluster Name' (set to 'MyCluster'), 'Databricks Runtime Version' (set to 'Runtime: 5.5 LTS (Scala 2.11, Spark 2.4.3)'), 'Python Version' (set to '3'), and an 'Instance' section with a note about free memory and upgrade options. At the bottom, tabs for 'Instances' (selected) and 'Spark' are visible, along with an 'Availability Zone' dropdown set to 'us-west-2c'. A red box highlights the 'Create Cluster' button and its tooltip information: '0 Workers: 0.0 GB Memory, 0 Cores, 0 DBU' and '1 Driver: 6.0 GB Memory, 0.88 Cores, 1 DBU'.

Create Cluster

New Cluster | Cancel | Create Cluster

0 Workers: 0.0 GB Memory, 0 Cores, 0 DBU  
1 Driver: 6.0 GB Memory, 0.88 Cores, 1 DBU

Cluster Name

MyCluster

Databricks Runtime Version

Runtime: 5.5 LTS (Scala 2.11, Spark 2.4.3)

Python Version

3

Instance

Free 6GB 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](#).

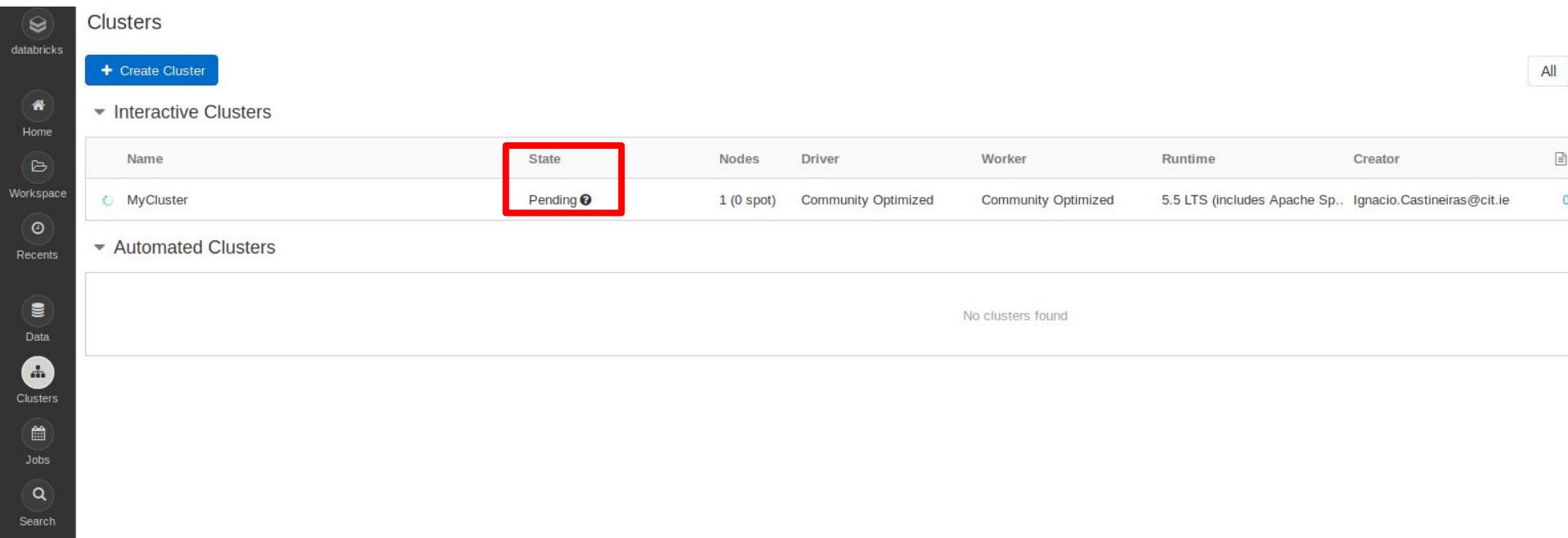
Instances | Spark

Availability Zone

us-west-2c

# Databricks: An Online Platform for Data Engineers

3. The cluster will take a couple of minutes to be set up:



The screenshot shows the Databricks interface with the 'Clusters' sidebar selected. A red box highlights the 'State' column for the 'MyCluster' entry, which is currently 'Pending'. The table also displays columns for Name, Nodes, Driver, Worker, Runtime, and Creator.

Name	State	Nodes	Driver	Worker	Runtime	Creator
MyCluster	Pending ?	1 (0 spot)	Community Optimized	Community Optimized	5.5 LTS (includes Apache Sp..)	Ignacio.Castineiras@cit.ie

# Databricks: An Online Platform for Data Engineers

3. The cluster will take a couple of minutes to be set up:

The screenshot shows the Databricks interface with the 'Clusters' tab selected. On the left, there is a sidebar with icons for Home, Workspace, Recents, Data, Clusters (which is selected), Jobs, and Search. The main area is titled 'Clusters' and shows a table of clusters. A blue button at the top left says '+ Create Cluster'. Under 'Interactive Clusters', there is one entry: 'MyCluster' (indicated by a green dot) with a status of 'Running'. The table columns are Name, State, Nodes, Driver, Worker, Runtime, and Creator. The 'State' column for MyCluster is highlighted with a red box. Below the table, under 'Automated Clusters', it says 'No clusters found'. At the top right, there is a 'All' dropdown menu.

Name	State	Nodes	Driver	Worker	Runtime	Creator
MyCluster	Running	1 (0 spot)	Community Optimized	Community Optimized	5.5 LTS (includes Apache Sp..	Ignacio.Castineiras@cit.ie

# Databricks: An Online Platform for Data Engineers

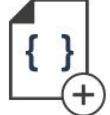
3. We can come back to the main page of the web interface by clicking in **databricks**.



Welcome to  databricks™

  
[Explore the Quickstart Tutorial](#)  
Spin up a cluster, run queries on preloaded data, and display results in 5 minutes.

  
[Import & Explore Data](#)  
Quickly import data, preview its schema, create a table, and query it in a notebook.

  
[Create a Blank Notebook](#)  
Create a notebook to start querying, visualizing, and modeling your data.

**Common Tasks**

-  New Notebook
-  Create Table
-  New Cluster

**Recents**

-  34\_job\_inspection\_5.scala
-  33\_job\_inspection\_4.scala
-  32\_job\_inspection\_3.scala

**What's new in v3.2**

- Instance Pools
- Databricks Runtime 6.0 will drop Python 2 support

[View latest release notes](#)

# Databricks: An Online Platform for Data Engineers

*How to...*

Create a new program / Spark Application.

# Databricks: An Online Platform for Data Engineers

4. Our next step is to create a new Spark application as a new notebook in our **Workspace**.



Welcome to  databricks™

The workspace interface features three main sections:

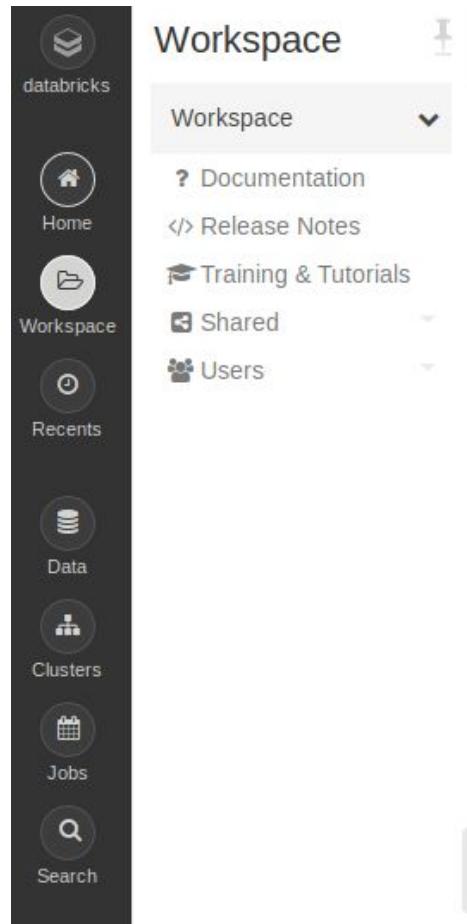
- Explore the Quickstart Tutorial**: Spins up a cluster, runs queries on preloaded data, and displays results in 5 minutes.
- Import & Explore Data**: Imports data, previews its schema, creates a table, and queries it in a notebook.
- Create a Blank Notebook**: Creates a new notebook to start querying, visualizing, and modeling your data.

Below these sections are three tabs: **Common Tasks**, **Recents**, and **What's new in v3.2**.

- Common Tasks** includes: New Notebook, Create Table, and New Cluster.
- Recents** lists recent files: 34\_job\_inspection\_5.scala, 33\_job\_inspection\_4.scala, and 32\_job\_inspection\_3.scala.
- What's new in v3.2** highlights: Instance Pools and Databricks Runtime 6.0 will drop Python 2 support. A link to View latest release notes is also provided.

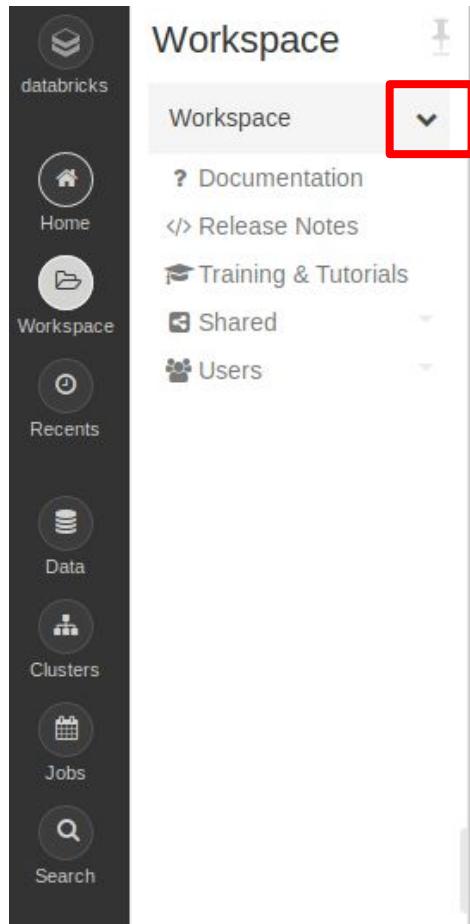
# Databricks: An Online Platform for Data Engineers

4. Our next step is to create a new Spark application as a new notebook in our **Workspace**.



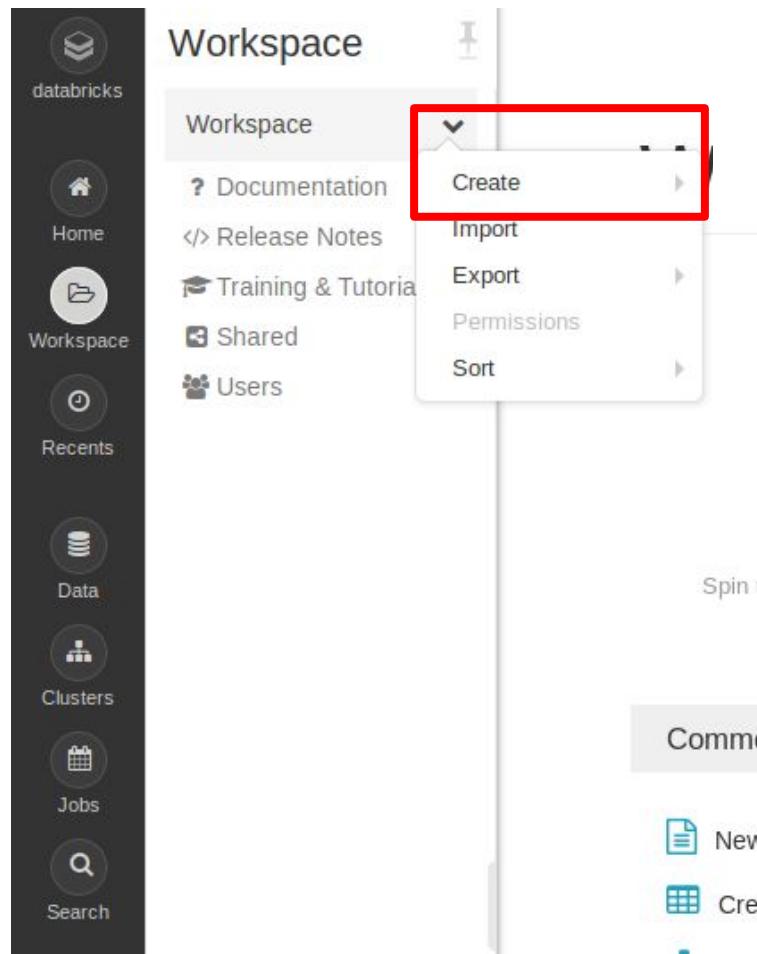
# Databricks: An Online Platform for Data Engineers

4. Our next step is to create a new Spark application as a new notebook in our **Workspace**.



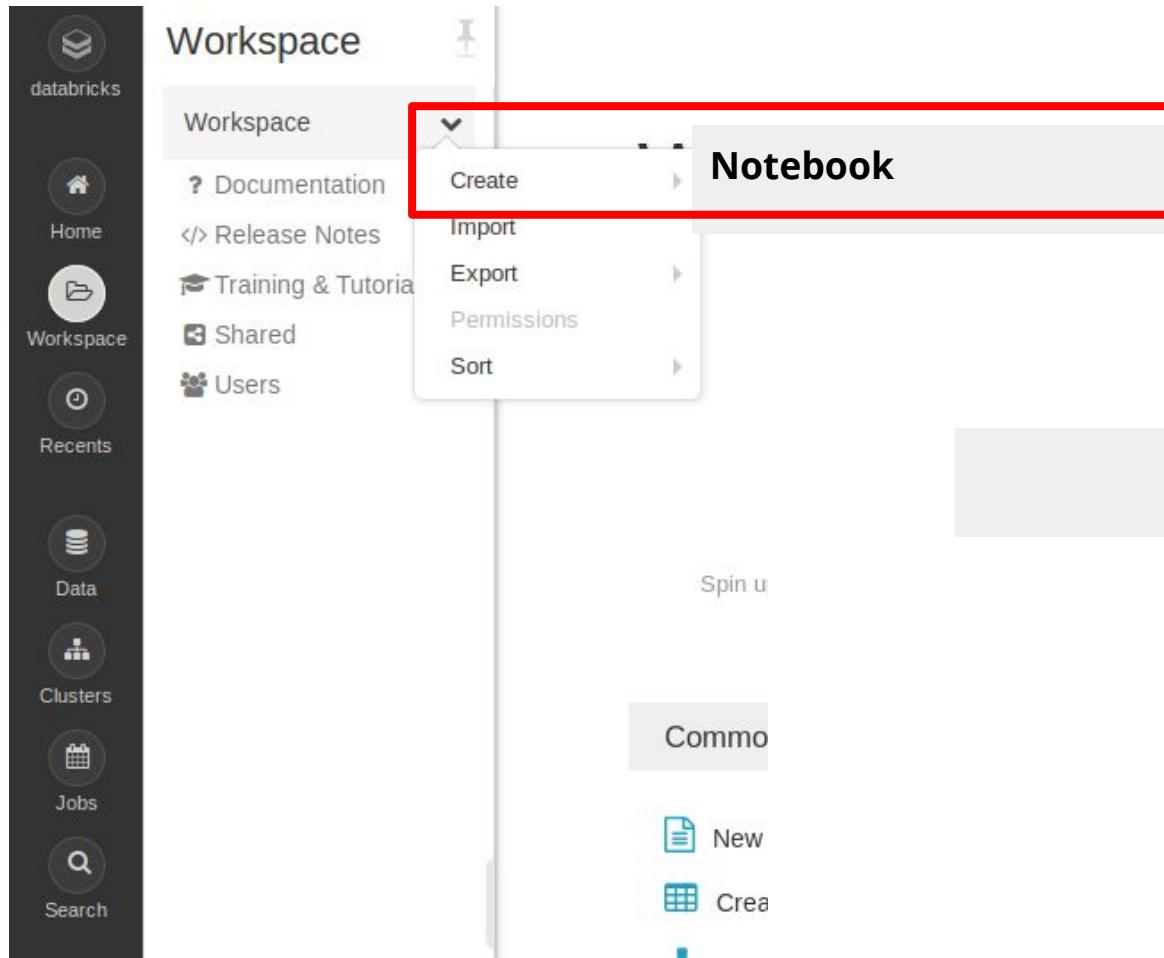
# Databricks: An Online Platform for Data Engineers

4. Our next step is to create a new Spark application as a new notebook in our **Workspace**.



# Databricks: An Online Platform for Data Engineers

4. Our next step is to create a new Spark application as a new notebook in our **Workspace**.



# Databricks: An Online Platform for Data Engineers

4. Our next step is to create a new Spark application as a new notebook in our **Workspace**.

The screenshot shows the Databricks workspace interface. On the left, there is a sidebar with various icons for Workspace, Home, Workspace, Training & Tutorials, Shared, Users, Data, Clusters, Jobs, and Search. The main area displays a "Welcome to databricks™" message with a "Create Notebook" button. A modal dialog box titled "Create Notebook" is open, containing fields for "Name" (set to "my\_first\_program.py"), "Language" (set to "Python"), and "Cluster" (set to "MyCluster"). At the bottom right of the dialog are "Cancel" and "Create" buttons. Below the dialog, there are sections for "Common Tasks" (with "New Notebook" and "Create Table" options) and "Recents" (with a placeholder message: "Recent files appear here as you work").

# Databricks: An Online Platform for Data Engineers

4. Our next step is to create a new Spark application as a new notebook in our **Workspace**.

The screenshot shows the Databricks workspace interface. On the left, there is a sidebar with various icons for Home, Workspace, Recents, Data, Clusters, Jobs, and Search. The main area displays a "Welcome to databricks™" message with a "Create Notebook" button. A modal dialog box is open, titled "Create Notebook". It contains three input fields: "Name" (with the value "my\_first\_program.py" highlighted by a red box), "Language" (set to "Python"), and "Cluster" (set to "MyCluster"). At the bottom right of the dialog are "Cancel" and "Create" buttons.

Workspace

Workspace

Documentation

Release Notes

Training & Tutorials

Shared

Users

Welcome to databricks™

Create Notebook

Name: my\_first\_program.py

Language: Python

Cluster: MyCluster

Explore the Quickstart Tutorial

Spin up a cluster, run queries on preloaded data, and see results in 5 minutes.

Common Tasks

New Notebook

Create Table

Recents

Recent files appear here as you work.

# Databricks: An Online Platform for Data Engineers

4. Our next step is to create a new Spark application as a new notebook in our **Workspace**.

The screenshot shows the Databricks workspace interface. On the left, there is a sidebar with various icons for Home, Workspace, Data, Clusters, Jobs, and Search. The main area displays a "Welcome to databricks" message with a "Create Notebook" button. A modal window titled "Create Notebook" is open, prompting for "Name" (set to "my\_first\_program.py"), "Language" (set to "Python" and highlighted with a red box), and "Cluster" (set to "MyCluster"). At the bottom right of the modal are "Cancel" and "Create" buttons.

Workspace

Workspace

Documentation

Release Notes

Training & Tutorials

Shared

Users

Welcome to databricks

Create Notebook

Name: my\_first\_program.py

Language: Python

Cluster: MyCluster

Explore the Quickstart Tutorial

Spin up a cluster, run queries on preloaded data, and see results in 5 minutes.

Common Tasks

New Notebook

Create Table

Recents

Recent files appear here as you work.

# Databricks: An Online Platform for Data Engineers

4. Our next step is to create a new Spark application as a new notebook in our **Workspace**.

The screenshot shows the Databricks workspace interface. On the left, there is a sidebar with various icons for Workspace, Home, Training & Tutorials, Shared, Users, Recents, Data, Clusters, Jobs, and Search. The main area displays a "Welcome to databricks™" message with a "Create Notebook" button. A modal dialog box titled "Create Notebook" is open, prompting for a "Name" (set to "my\_first\_program.py"), "Language" (set to "Python"), and "Cluster" (set to "MyCluster"). The "Cluster" field is highlighted with a red box. At the bottom right of the dialog are "Cancel" and "Create" buttons.

# Databricks: An Online Platform for Data Engineers

4. Our next step is to create a new Spark application as a new notebook in our **Workspace**.

The screenshot shows the Databricks workspace interface. On the left, there is a sidebar with various navigation icons: Home, Workspace, Recents, Data, Clusters, Jobs, and Search. The 'Workspace' icon is selected, and the 'Workspace' menu is open, showing options like Documentation, Release Notes, Training & Tutorials, Shared, and Users.

The main area features a 'Welcome to databricks™' banner with a 'Create Notebook' button. Below it is a 'Quickstart Tutorial' section with a 'Spin up a cluster...' button. At the bottom, there are 'Common Tasks' (New Notebook, Create Table) and 'Recents' sections.

A modal dialog box titled 'Create Notebook' is displayed in the center. It contains fields for 'Name' (set to 'my\_first\_program.py'), 'Language' (set to 'Python'), and 'Cluster' (set to 'MyCluster'). A 'Cancel' button and a 'Create' button are at the bottom right. The 'Create' button is highlighted with a red box.

# Databricks: An Online Platform for Data Engineers

4. We are redirected to the main web page to edit our notebook.  
At the beginning it is empty.

The screenshot shows the Databricks workspace interface. On the left is a sidebar with icons for Home, Workspace, Recents, Data, Clusters, Jobs, and Search. The main area displays a notebook titled "my\_first\_program.py (Python)". The notebook has one command cell, "Cmd 1", which contains the number "1". Below the cell is the instruction "Shift+Enter to run" and a link to "shortcuts". The top navigation bar includes tabs for "File", "View: Code", "Permissions", "Run All", "Clear", and buttons for "Publish", "Comments", "Runs", and "Revision history".

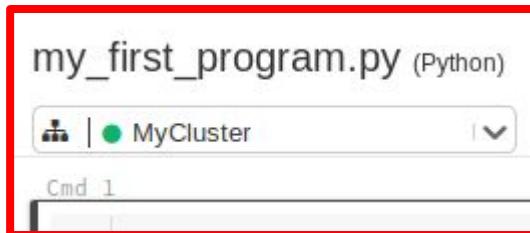
# Databricks: An Online Platform for Data Engineers

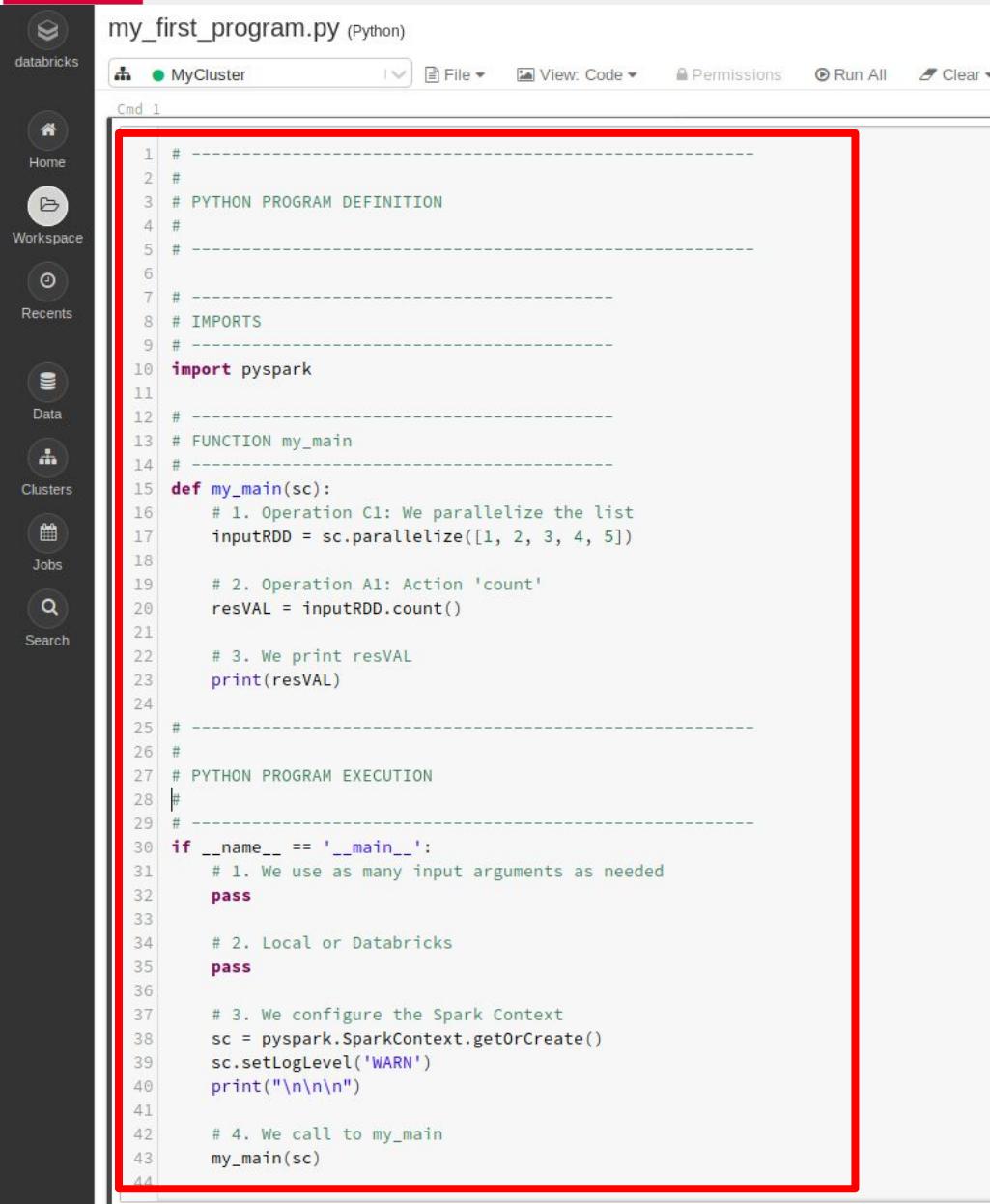
4. We are redirected to the main web page to edit our notebook.  
As we can see our program called **my\_first\_program.py** is automatically attached to our cluster **MyCluster**. Otherwise we can attach it by ourselves.

The screenshot shows the Databricks workspace interface. On the left is a sidebar with icons for Home, Workspace, Recents, Data, Clusters, Jobs, and Search. The main area displays a notebook titled "my\_first\_program.py (Python)". Above the code editor, there is a dropdown menu showing "MyCluster" selected. The top navigation bar includes File, View, Permissions, Run All, Clear, Publish, Comments, Runs, and Revision history. The code editor itself is currently empty, with the placeholder "Shift+Enter to run" visible.

# Databricks: An Online Platform for Data Engineers

4. We are redirected to the main web page to edit our notebook.  
As we can see our program called **my\_first\_program.py** is automatically attached to our cluster **MyCluster**. Otherwise we can attach it by ourselves.





my\_first\_program.py (Python)

MyCluster

File View: Code Permissions Run All Clear

Cmd 1

```
1 # -----
2 #
3 # PYTHON PROGRAM DEFINITION
4 #
5 # -----
6 #
7 # -----
8 # IMPORTS
9 #
10 import pyspark
11 #
12 # -----
13 # FUNCTION my_main
14 #
15 def my_main(sc):
16     # 1. Operation C1: We parallelize the list
17     inputRDD = sc.parallelize([1, 2, 3, 4, 5])
18 #
19     # 2. Operation A1: Action 'count'
20     resVAL = inputRDD.count()
21 #
22     # 3. We print resVAL
23     print(resVAL)
24 #
25 # -----
26 #
27 # PYTHON PROGRAM EXECUTION
28 #
29 #
30 if __name__ == '__main__':
31     # 1. We use as many input arguments as needed
32     pass
33 #
34     # 2. Local or Databricks
35     pass
36 #
37     # 3. We configure the Spark Context
38     sc = pyspark.SparkContext.getOrCreate()
39     sc.setLogLevel('WARN')
40     print("\n\n\n")
41 #
42     # 4. We call to my_main
43     my_main(sc)
44
```

4. We fill in our program in the Jupiter Notebook being created.

The screenshot shows the Databricks workspace interface. On the left is a sidebar with icons for Home, Workspace, Recents, Data, Clusters, Jobs, and Search. The main area is titled "my\_first\_program.py (Python)". It shows a single code cell labeled "Cmd 1" containing the following Python code:

```
1 # -----
2 #
3 # PYTHON PROGRAM DEFINITION
4 #
5 # -----
6
7 # -----
8 # IMPORTS
9 #
10 import pyspark
11
12 # -----
13 # FUNCTION my_main
14 #
15 def my_main(sc):
16     # 1. Operation C1: We parallelize the list
17     inputRDD = sc.parallelize([1, 2, 3, 4, 5])
18
19     # 2. Operation A1: Action 'count'
20     resVAL = inputRDD.count()
21
22     # 3. We print resVAL
23     print(resVAL)
24
25 #
26 #
27 # PYTHON PROGRAM EXECUTION
28 #
29 #
30 if __name__ == '__main__':
31     # 1. We use as many input arguments as needed
32     pass
33
34     # 2. Local or Databricks
35     pass
36
37     # 3. We configure the Spark Context
38     sc = pyspark.SparkContext.getOrCreate()
39     sc.setLogLevel('WARN')
40     print("\n\n\n")
41
42     # 4. We call to my_main
43     my_main(sc)
```

4. We fill in our program in the Jupiter Notebook being created.

### Note:

As not being a big fan of Jupiter Notebooks, let's please treat them as if they were regular Python programs.

That is, let's fill the entire Python program into the **Cmd 1** cell of the notebook.

# Databricks: An Online Platform for Data Engineers



The screenshot shows the Databricks workspace interface. On the left, there's a sidebar with icons for 'databricks' (home), 'Home', and 'Workspace'. The main area displays a Python notebook titled 'my\_first\_program.py (Python)'. The notebook has one cell named 'Cmd 1', which contains the following code:

```
1 # -----
2 #
3 # PYTHON PROGRAM DEFINITION
4 #
5 # -----
```

A red box highlights the 'Cmd 1' cell.

4. We fill in our program in the Jupiter Notebook being created.

Note:

As not being a big fan of Jupiter Notebooks, let's please treat them as if they were regular Python programs.

That is, let's fill the entire Python program into the **Cmd 1** cell of the notebook.

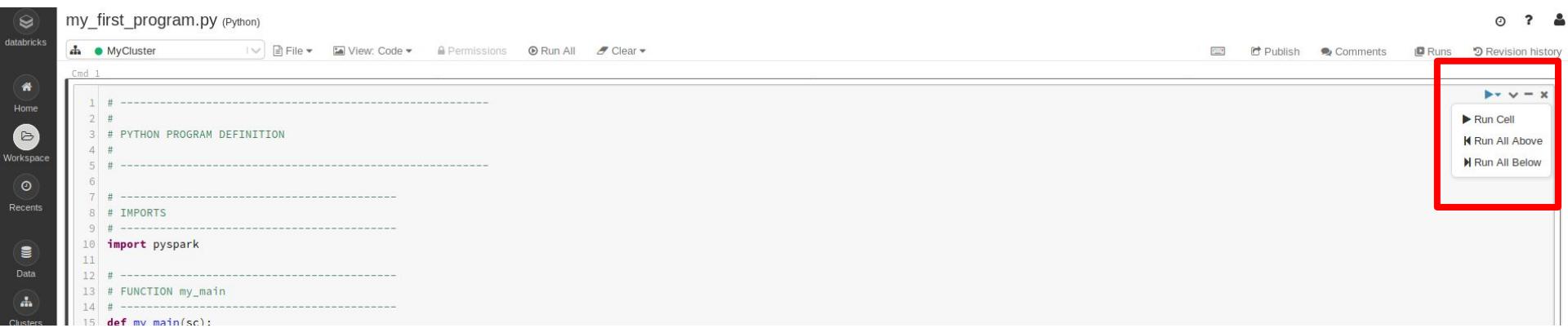
# Databricks: An Online Platform for Data Engineers

*How to...*

Run the new program / Spark Application.

# Databricks: An Online Platform for Data Engineers

- Once the program (Spark application) is ready, we can execute it in our cluster.



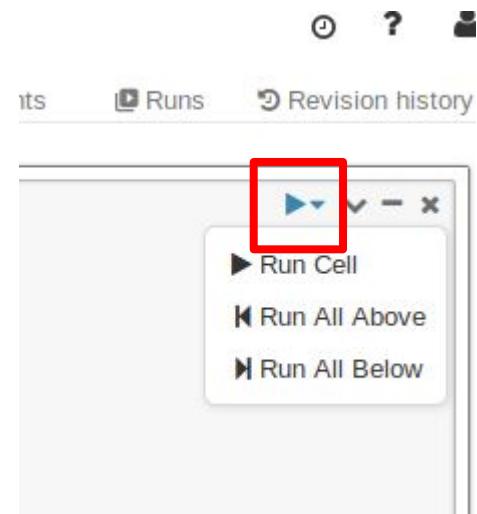
The screenshot shows the Databricks workspace interface. On the left, there's a sidebar with icons for Home, Workspace, Recents, Data, and Clusters. The main area displays a Python notebook titled "my\_first\_program.py". The code in the notebook is:

```
1 # -----
2 #
3 # PYTHON PROGRAM DEFINITION
4 #
5 #
6 #
7 # -----
8 # IMPORTS
9 #
10 import pyspark
11 #
12 # -----
13 # FUNCTION my_main
14 #
15 def my_main(sc):
```

At the top of the notebook, there are tabs for "MyCluster" and "File". Below the tabs are buttons for "View: Code", "Permissions", "Run All", "Clear", "Publish", "Comments", "Runs", and "Revision history". On the right side of the notebook, there's a panel with three run control buttons: "Run Cell", "Run All Above", and "Run All Below". A red box highlights this panel.

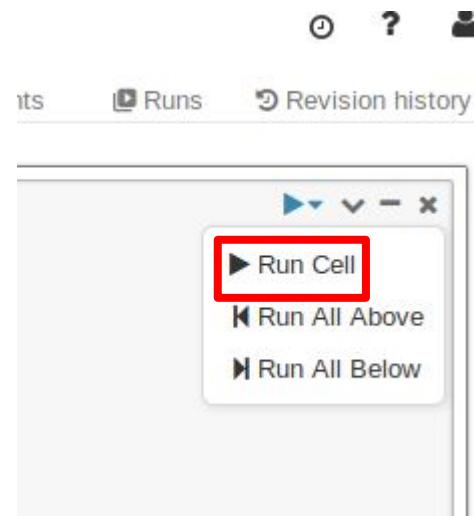
# Databricks: An Online Platform for Data Engineers

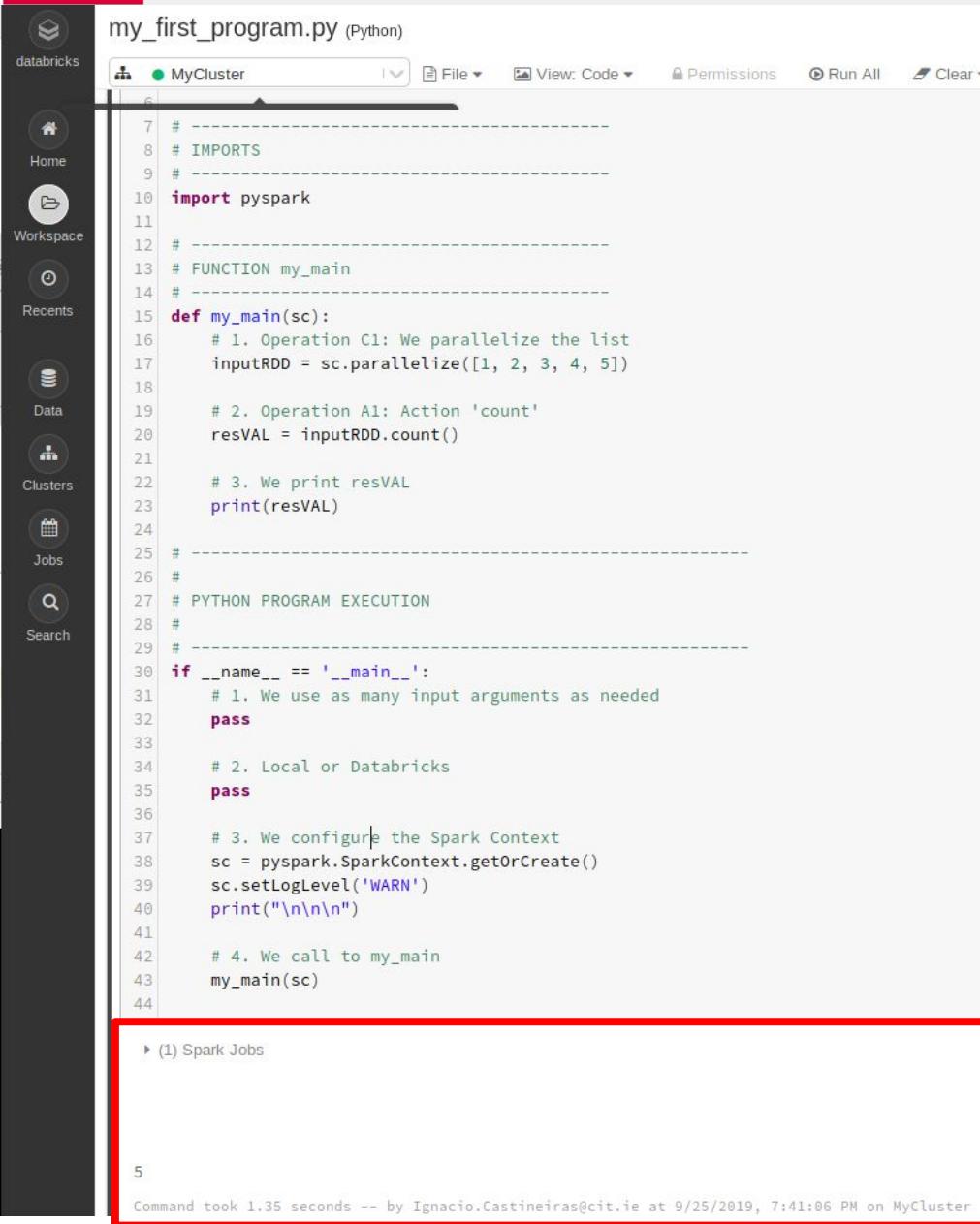
- Once the program (Spark application) is ready, we can execute it in our cluster.



# Databricks: An Online Platform for Data Engineers

- Once the program (Spark application) is ready, we can execute it in our cluster.





my\_first\_program.py (Python)

MyCluster

File View: Code Permissions Run All Clear

```
6
7 # -----
8 # IMPORTS
9 # -----
10 import pyspark
11
12 # -----
13 # FUNCTION my_main
14 #
15 def my_main(sc):
16     # 1. Operation C1: We parallelize the list
17     inputRDD = sc.parallelize([1, 2, 3, 4, 5])
18
19     # 2. Operation A1: Action 'count'
20     resVAL = inputRDD.count()
21
22     # 3. We print resVAL
23     print(resVAL)
24
25 # -----
26 #
27 # PYTHON PROGRAM EXECUTION
28 #
29 #
30 if __name__ == '__main__':
31     # 1. We use as many input arguments as needed
32     pass
33
34     # 2. Local or Databricks
35     pass
36
37     # 3. We configure the Spark Context
38     sc = pyspark.SparkContext.getOrCreate()
39     sc.setLogLevel('WARN')
40     print("\n\n\n")
41
42     # 4. We call to my_main
43     my_main(sc)
44
```

(1) Spark Jobs

5

Command took 1.35 seconds -- by Ignacio.Castineiras@cit.ie at 9/25/2019, 7:41:06 PM on MyCluster

And we can evaluate its result.

# Databricks: An Online Platform for Data Engineers

5. And we can evaluate its result.

```
34      # 2. Local or Databricks
35      pass
36
37      # 3. We configure the Spark Context
38      sc = pyspark.SparkContext.getOrCreate()
39      sc.setLogLevel('WARN')
40      print("\n\n\n")
41
42      # 4. We call to my_main
43      my_main(sc)
44
```

▶ (1) Spark Jobs

5

Command took 1.35 seconds -- by Ignacio.Castineiras@cit.ie at 9/25/2019, 7:41:06 PM on MyCluster

# Databricks: An Online Platform for Data Engineers

5. And we can evaluate its result.

```
34      # 2. Local or Databricks
35      pass
36
37      # 3. We configure the Spark Context
38      sc = pyspark.SparkContext.getOrCreate()
39      sc.setLogLevel('WARN')
40      print("\n\n\n")
41
42      # 4. We call to my_main
43      my_main(sc)
44
```

▶ (1) Spark Jobs

5

Command took 1.35 seconds -- by Ignacio.Castineiras@cit.ie at 9/25/2019, 7:41:06 PM on MyCluster

# Databricks: An Online Platform for Data Engineers

5. And we can evaluate its result.

```
34      # 2. Local or Databricks
35      pass
36
37      # 3. We configure the Spark Context
38      sc = pyspark.SparkContext.getOrCreate()
39      sc.setLogLevel('WARN')
40      print("\n\n\n")
41
42      # 4. We call to my_main
43      my_main(sc)
44
```

▶ (1) Spark Jobs

5

Command took 1.35 seconds -- by Ignacio.Castineiras@cit.ie at 9/25/2019, 7:41:06 PM on MyCluster

# Databricks: An Online Platform for Data Engineers

5. And we can evaluate its result.

```
34     # 2. Local or Databricks
35     pass
36
37     # 3. We configure the Spark Context
38     sc = pyspark.SparkContext.getOrCreate()
39     sc.setLogLevel('WARN')
40     print("\n\n\n")
41
42     # 4. We call to my_main
43     my_main(sc)
44
```

▼ (1) Spark Jobs  
▶ Job 0 [View](#) (Stages: 1/1)

# Databricks: An Online Platform for Data Engineers

5. And we can evaluate its result.

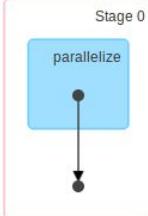
This includes inspecting the DAG and the Task Scheduling.

Jobs Stages Storage Environment Executors SQL JDBC/ODBC Server  

### Details for Job 0

Status: SUCCEEDED  
Job Group: 8217793805788340145\_5480434077039448585\_6284f447237e4486984fe6cd5a737250  
Completed Stages: 1

► Event Timeline  
▼DAG Visualization



Stage 0

parallelize

▼Completed Stages (1)

Stage Id	Pool Name	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
0	8217793805788340145	# ----- count at <command-3580226803360605>:20 +details	2019/09/26 14:17:32	2 s	8/8				

# Databricks: An Online Platform for Data Engineers

5. And we can evaluate its result.

This includes inspecting the **DAG** and the Task Scheduling.

The screenshot shows the Databricks interface for 'Job 0'. The top navigation bar includes tabs for Jobs, Stages, Storage, Environment, Executors, SQL, and JDBC/ODBC Server. Below the navigation is a toolbar with a refresh icon and a close button. The main content area displays 'Details for Job 0' with a status of 'SUCCEEDED'. It shows the 'Job Group' ID and 'Completed Stages: 1'. A red box highlights the 'DAG Visualization' section, which contains a diagram for 'Stage 0' showing a single task named 'parallelize'. Below the DAG is a table titled 'Completed Stages (1)' with one row of data.

Stage Id	Pool Name	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
0	8217793805788340145	# ----- count at <Command-3580226803360605>:20 +details	2019/09/26 14:17:32	2 s	8/8				

# Databricks: An Online Platform for Data Engineers

5. And we can evaluate its result.

This includes inspecting the DAG and the **Task Scheduling**.

Jobs Stages Storage Environment Executors SQL JDBC/ODBC Server

Details for Job 0

Status: SUCCEEDED  
Job Group: 8217793805788340145\_5480434077039448585\_6284f447237e4486984fe6cd5a737250  
Completed Stages: 1

► Event Timeline  
▼DAG Visualization

Stage 0

```
graph TD; A[parallelize] --> B(( ));
```

▼Completed Stages (1)

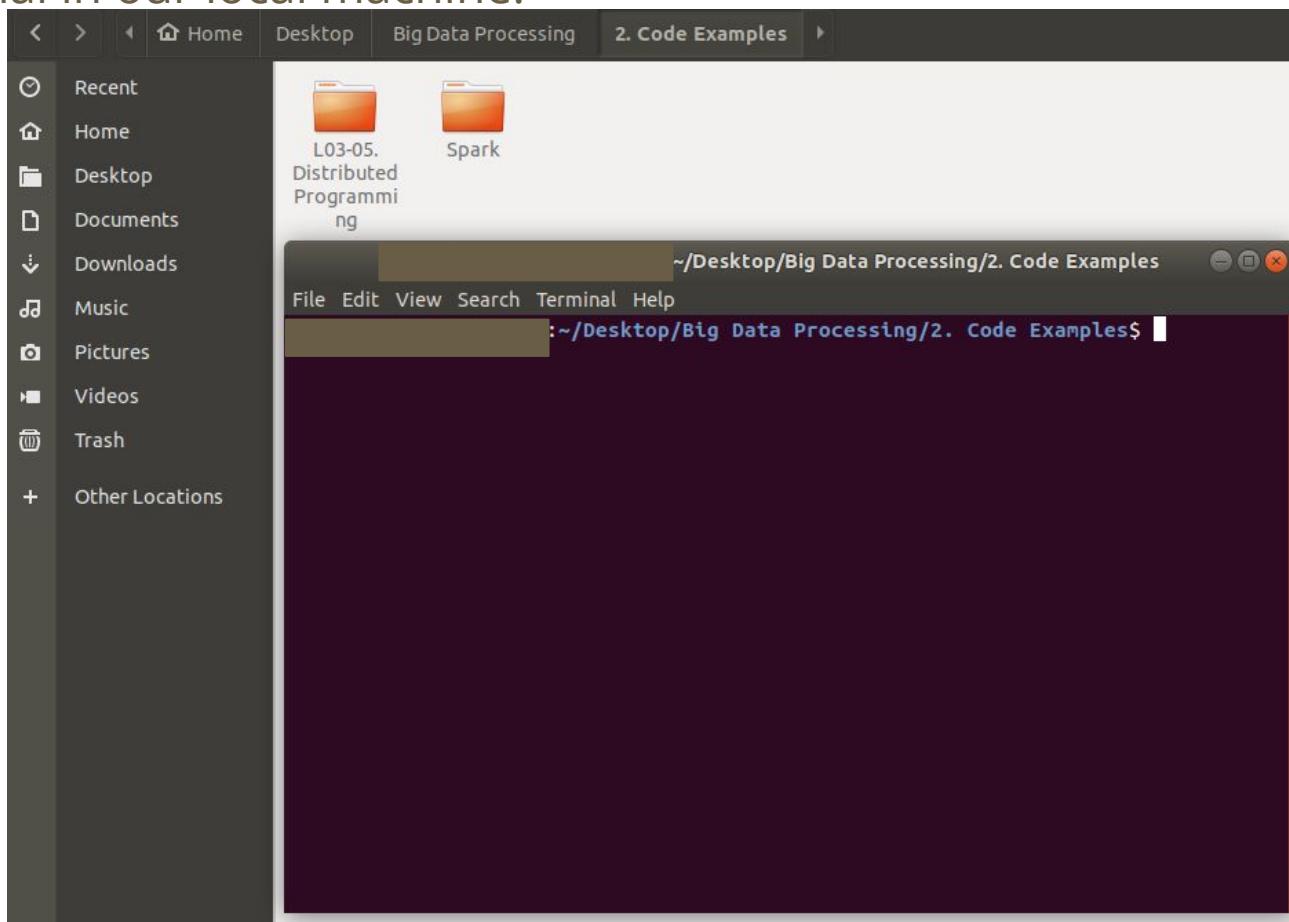
Stage Id	Pool Name	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
0	8217793805788340145	# ----- count at <command-3580226803360605>:20 +details	2019/09/26 14:17:32	2 s	8/8				

# Databricks: An Online Platform for Data Engineers

*How to...*  
Install the Databricks  
Command Line Application (CLI)

# Databricks: An Online Platform for Data Engineers

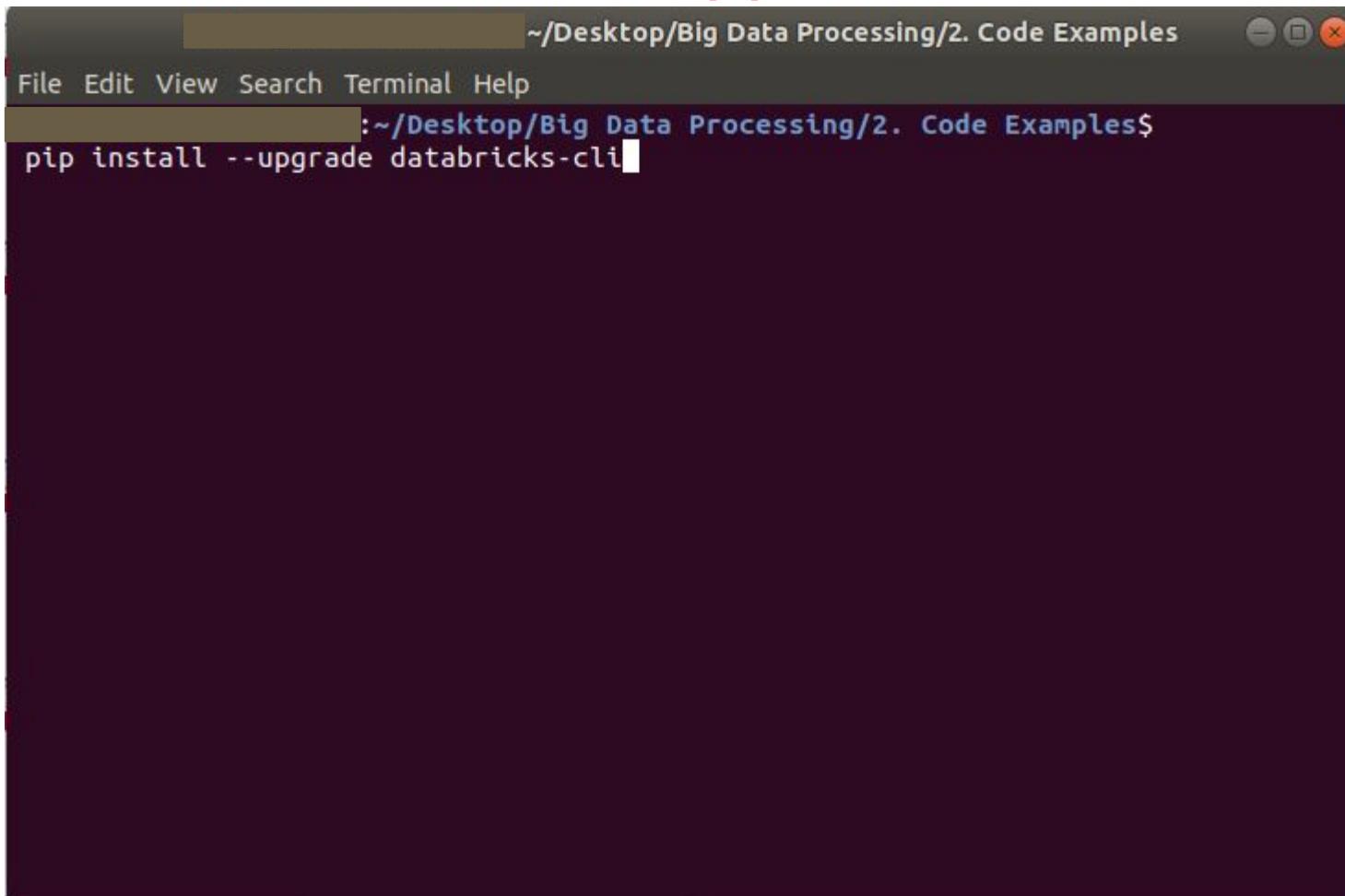
6. Databricks provides a Command Line Interface (CLI).  
The CLI allows us to perform some operations on Databricks from a terminal in our local machine.



# Databricks: An Online Platform for Data Engineers

6. Databricks provides a Command Line Interface (CLI).

We can install the Databricks CLI via **pip**.

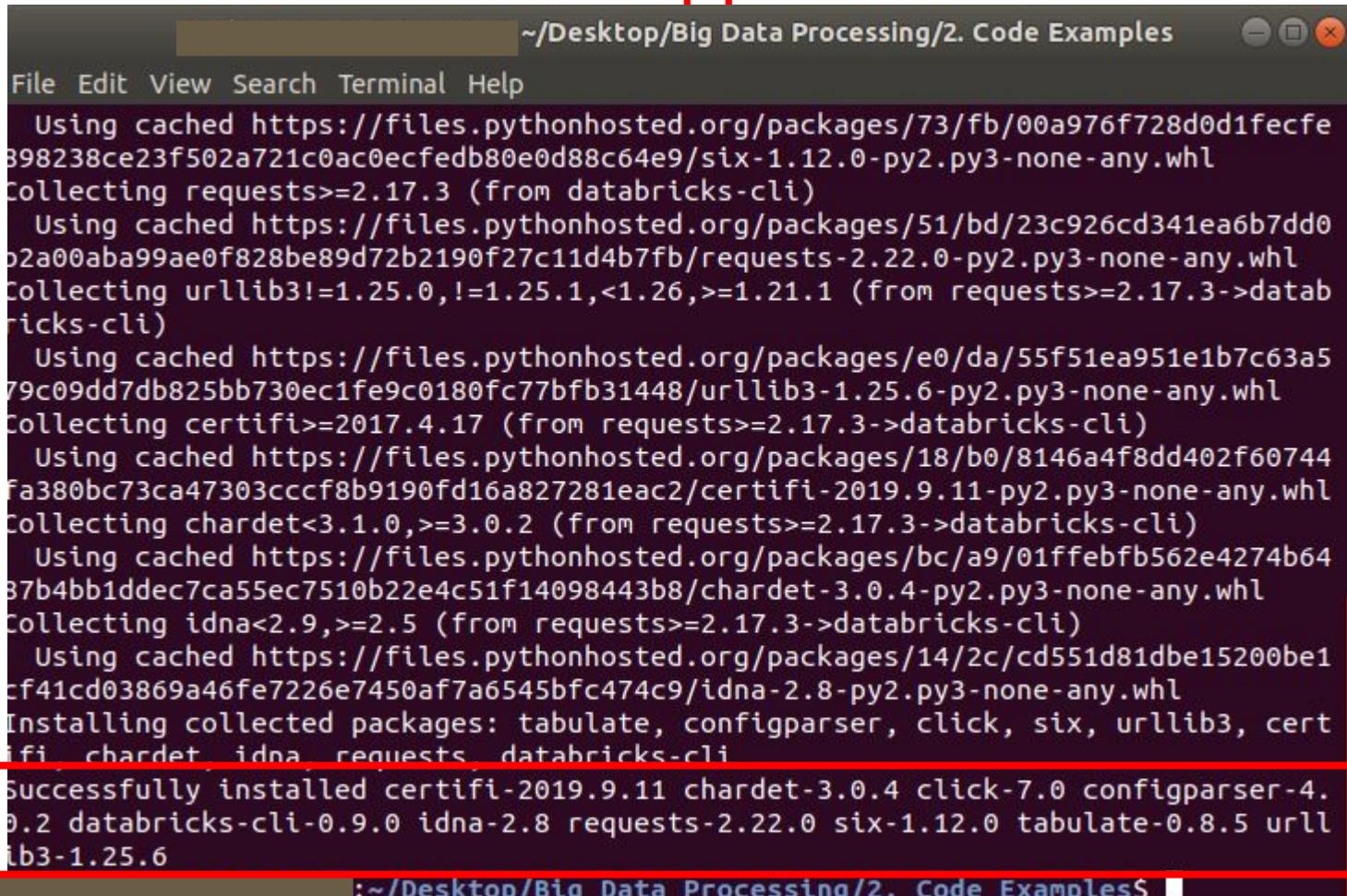


A screenshot of a terminal window titled '~/Desktop/Big Data Processing/2. Code Examples'. The window has a dark background and light-colored text. At the top, there is a menu bar with 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. Below the menu, the current directory is shown as ':~/Desktop/Big Data Processing/2. Code Examples\$'. A command is being typed into the terminal: 'pip install --upgrade databricks-cli'. The cursor is positioned at the end of the command.

# Databricks: An Online Platform for Data Engineers

6. Databricks provides a Command Line Interface (CLI).

We can install the Databricks CLI via **pip**.



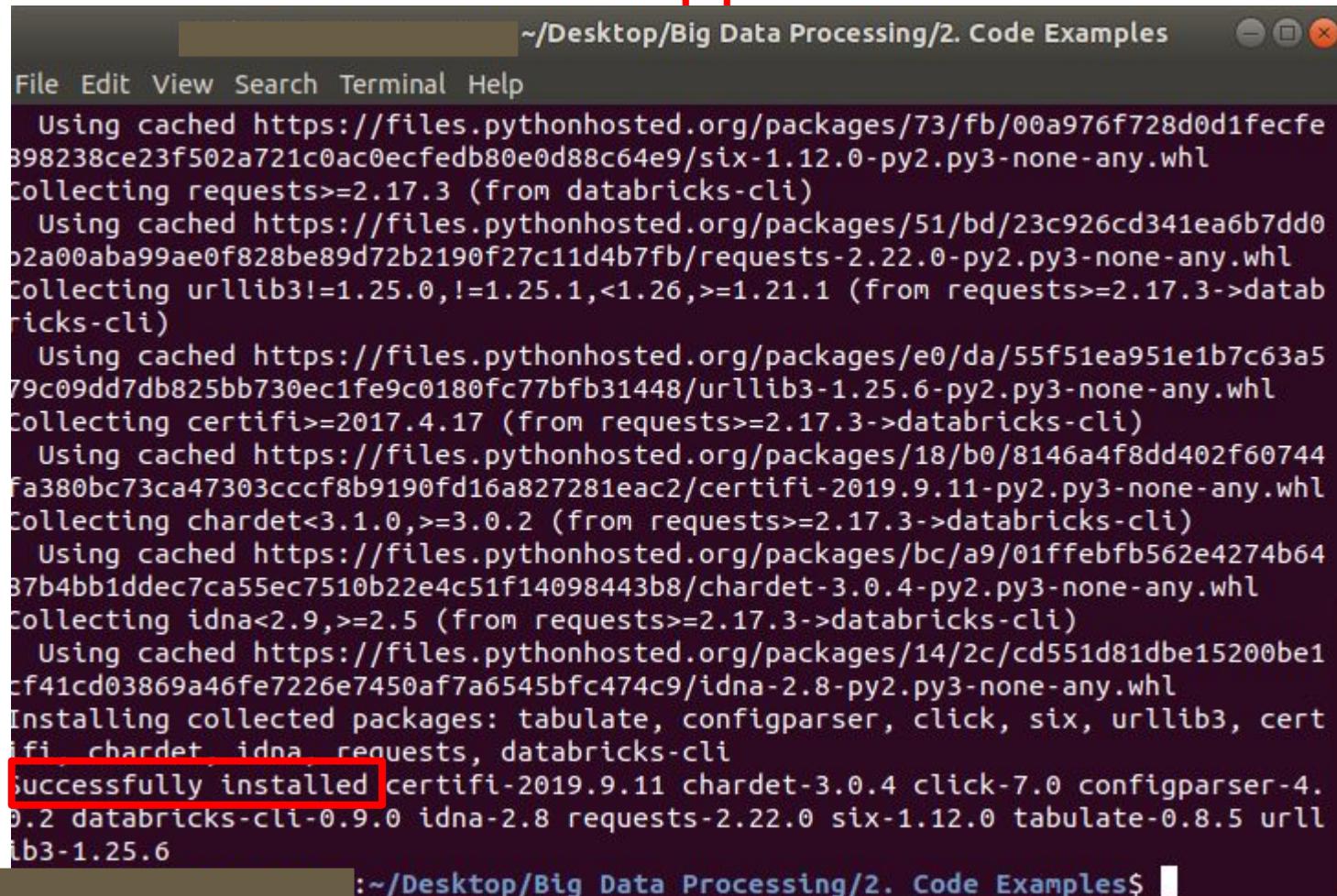
The screenshot shows a terminal window titled "/Desktop/Big Data Processing/2. Code Examples". The window contains the output of a pip command to install the Databricks CLI. The output shows the download and extraction of various Python packages from pythonhosted.org, including certifi, chardet, idna, requests, tabulate, configparser, click, six, urllib3, and databricks-cli. The final message indicates successful installation of all collected packages.

```
~/Desktop/Big Data Processing/2. Code Examples$ pip install databricks-cli
Using cached https://files.pythonhosted.org/packages/73/fb/00a976f728d0d1fecfe
398238ce23f502a721c0ac0ecfedb80e0d88c64e9/six-1.12.0-py2.py3-none-any.whl
Collecting requests>=2.17.3 (from databricks-cli)
  Using cached https://files.pythonhosted.org/packages/51/bd/23c926cd341ea6b7dd0
b2a00aba99ae0f828be89d72b2190f27c11d4b7fb/requests-2.22.0-py2.py3-none-any.whl
Collecting urllib3!=1.25.0,!>1.25.1,<1.26,>=1.21.1 (from requests>=2.17.3->datab
ricks-cli)
  Using cached https://files.pythonhosted.org/packages/e0/da/55f51ea951e1b7c63a5
79c09dd7db825bb730ec1fe9c0180fc77bfb31448/urllib3-1.25.6-py2.py3-none-any.whl
Collecting certifi>=2017.4.17 (from requests>=2.17.3->databricks-cli)
  Using cached https://files.pythonhosted.org/packages/18/b0/8146a4f8dd402f60744
fa380bc73ca47303cccf8b9190fd16a827281eac2/certifi-2019.9.11-py2.py3-none-any.whl
Collecting chardet<3.1.0,>=3.0.2 (from requests>=2.17.3->databricks-cli)
  Using cached https://files.pythonhosted.org/packages/bc/a9/01ffebfb562e4274b64
87b4bb1ddec7ca55ec7510b22e4c51f14098443b8/chardet-3.0.4-py2.py3-none-any.whl
Collecting idna<2.9,>=2.5 (from requests>=2.17.3->databricks-cli)
  Using cached https://files.pythonhosted.org/packages/14/2c/cd551d81dbe15200be1
cf41cd03869a46fe7226e7450af7a6545bfc474c9/idna-2.8-py2.py3-none-any.whl
Installing collected packages: tabulate, configparser, click, six, urllib3, cert
ifi, chardet, idna, requests, databricks-cli
Successfully installed certifi-2019.9.11 chardet-3.0.4 click-7.0 configparser-4.
0.2 databricks-cli-0.9.0 idna-2.8 requests-2.22.0 six-1.12.0 tabulate-0.8.5 url
lib3-1.25.6
~/Desktop/Big Data Processing/2. Code Examples$
```

# Databricks: An Online Platform for Data Engineers

6. Databricks provides a Command Line Interface (CLI).

We can install the Databricks CLI via **pip**.



```
~/Desktop/Big Data Processing/2. Code Examples
```

```
File Edit View Search Terminal Help
```

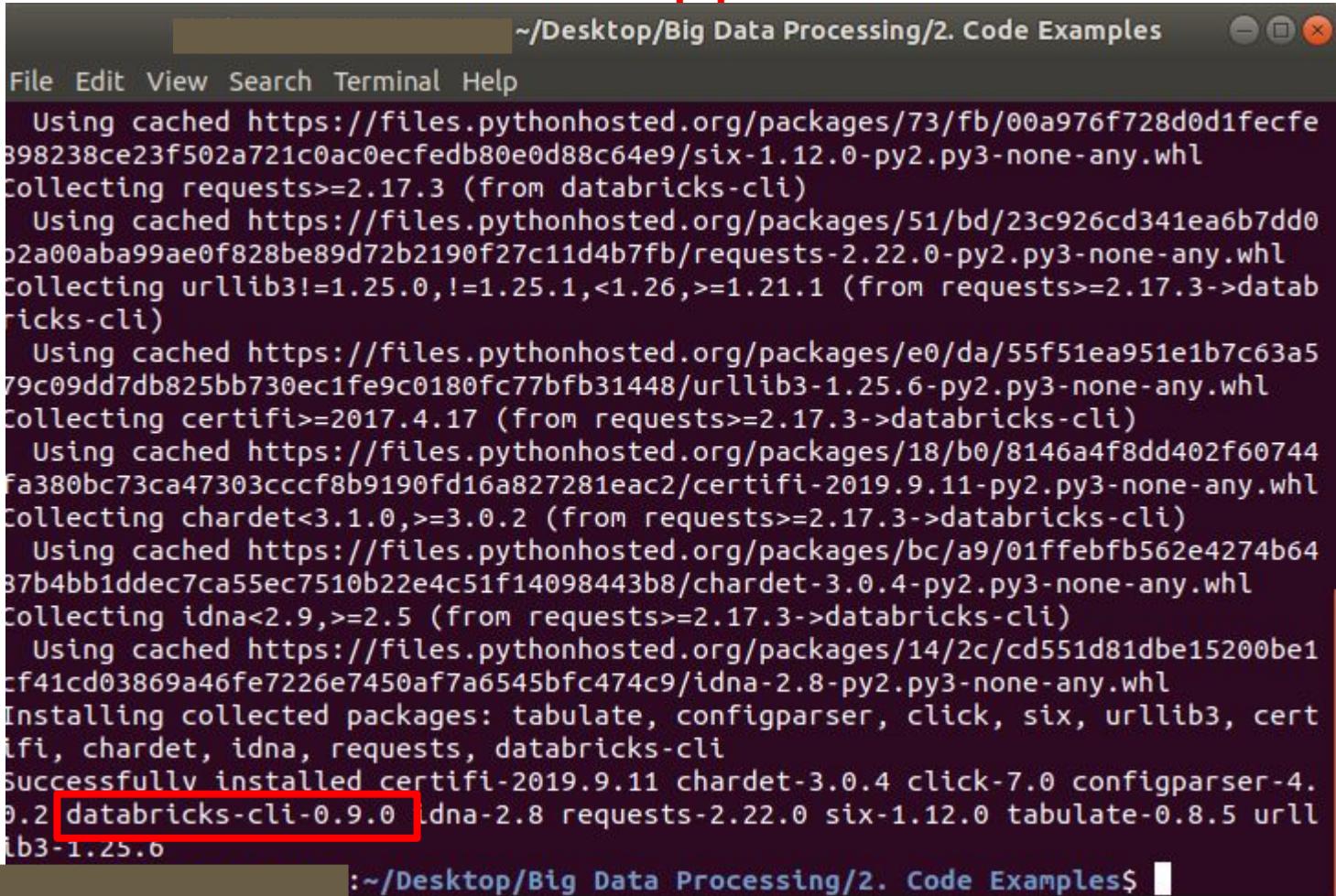
```
Using cached https://files.pythonhosted.org/packages/73/fb/00a976f728d0d1fecfe  
398238ce23f502a721c0ac0ecfedb80e0d88c64e9/six-1.12.0-py2.py3-none-any.whl  
Collecting requests>=2.17.3 (from databricks-cli)  
  Using cached https://files.pythonhosted.org/packages/51/bd/23c926cd341ea6b7dd0  
b2a00aba99ae0f828be89d72b2190f27c11d4b7fb/requests-2.22.0-py2.py3-none-any.whl  
Collecting urllib3!=1.25.0,!>1.25.1,<1.26,>=1.21.1 (from requests>=2.17.3->datab  
ricks-cli)  
  Using cached https://files.pythonhosted.org/packages/e0/da/55f51ea951e1b7c63a5  
79c09dd7db825bb730ec1fe9c0180fc77bfb31448/urllib3-1.25.6-py2.py3-none-any.whl  
Collecting certifi>=2017.4.17 (from requests>=2.17.3->databricks-cli)  
  Using cached https://files.pythonhosted.org/packages/18/b0/8146a4f8dd402f60744  
fa380bc73ca47303cccf8b9190fd16a827281eac2/certifi-2019.9.11-py2.py3-none-any.whl  
Collecting chardet<3.1.0,>=3.0.2 (from requests>=2.17.3->databricks-cli)  
  Using cached https://files.pythonhosted.org/packages/bc/a9/01ffebfb562e4274b64  
87b4bb1ddec7ca55ec7510b22e4c51f14098443b8/chardet-3.0.4-py2.py3-none-any.whl  
Collecting idna<2.9,>=2.5 (from requests>=2.17.3->databricks-cli)  
  Using cached https://files.pythonhosted.org/packages/14/2c/cd551d81dbe15200be1  
cf41cd03869a46fe7226e7450af7a6545bfc474c9/idna-2.8-py2.py3-none-any.whl  
Installing collected packages: tabulate, configparser, click, six, urllib3, cert  
ifi, chardet, idna, requests, databricks-cli  
Successfully installed certifi-2019.9.11 chardet-3.0.4 click-7.0 configparser-4.  
0.2 databricks-cli-0.9.0 idna-2.8 requests-2.22.0 six-1.12.0 tabulate-0.8.5 url  
ib3-1.25.6
```

```
~/Desktop/Big Data Processing/2. Code Examples$
```

# Databricks: An Online Platform for Data Engineers

6. Databricks provides a Command Line Interface (CLI).

We can install the Databricks CLI via **pip**.



The screenshot shows a terminal window titled "/Desktop/Big Data Processing/2. Code Examples". The terminal is displaying the output of a pip command to install the Databricks CLI. The output shows the process of collecting and installing various Python packages from pythonhosted.org, including certifi, chardet, idna, requests, and databricks-cli. The final line of the output, "Successfully installed databricks-cli-0.9.0", is highlighted with a red box.

```
~/Desktop/Big Data Processing/2. Code Examples$ pip install databricks-cli
Using cached https://files.pythonhosted.org/packages/73/fb/00a976f728d0d1fecfe
398238ce23f502a721c0ac0ecfedb80e0d88c64e9/six-1.12.0-py2.py3-none-any.whl
Collecting requests>=2.17.3 (from databricks-cli)
  Using cached https://files.pythonhosted.org/packages/51/bd/23c926cd341ea6b7dd0
b2a00aba99ae0f828be89d72b2190f27c11d4b7fb/requests-2.22.0-py2.py3-none-any.whl
Collecting urllib3!=1.25.0,!>1.25.1,<1.26,>=1.21.1 (from requests>=2.17.3->datab
ricks-cli)
  Using cached https://files.pythonhosted.org/packages/e0/da/55f51ea951e1b7c63a5
79c09dd7db825bb730ec1fe9c0180fc77bfb31448/urllib3-1.25.6-py2.py3-none-any.whl
Collecting certifi>=2017.4.17 (from requests>=2.17.3->databricks-cli)
  Using cached https://files.pythonhosted.org/packages/18/b0/8146a4f8dd402f60744
fa380bc73ca47303cccf8b9190fd16a827281eac2/certifi-2019.9.11-py2.py3-none-any.whl
Collecting chardet<3.1.0,>=3.0.2 (from requests>=2.17.3->databricks-cli)
  Using cached https://files.pythonhosted.org/packages/bc/a9/01ffebfb562e4274b64
87b4bb1ddec7ca55ec7510b22e4c51f14098443b8/chardet-3.0.4-py2.py3-none-any.whl
Collecting idna<2.9,>=2.5 (from requests>=2.17.3->databricks-cli)
  Using cached https://files.pythonhosted.org/packages/14/2c/cd551d81dbe15200be1
cf41cd03869a46fe7226e7450af7a6545bfc474c9/idna-2.8-py2.py3-none-any.whl
Installing collected packages: tabulate, configparser, click, six, urllib3, cert
ifi, chardet, idna, requests, databricks-cli
Successfully installed certifi-2019.9.11 chardet-3.0.4 click-7.0 configparser-4.
0.2 databricks-cli-0.9.0 idna-2.8 requests-2.22.0 six-1.12.0 tabulate-0.8.5 url
ib3-1.25.0
~/Desktop/Big Data Processing/2. Code Examples$
```

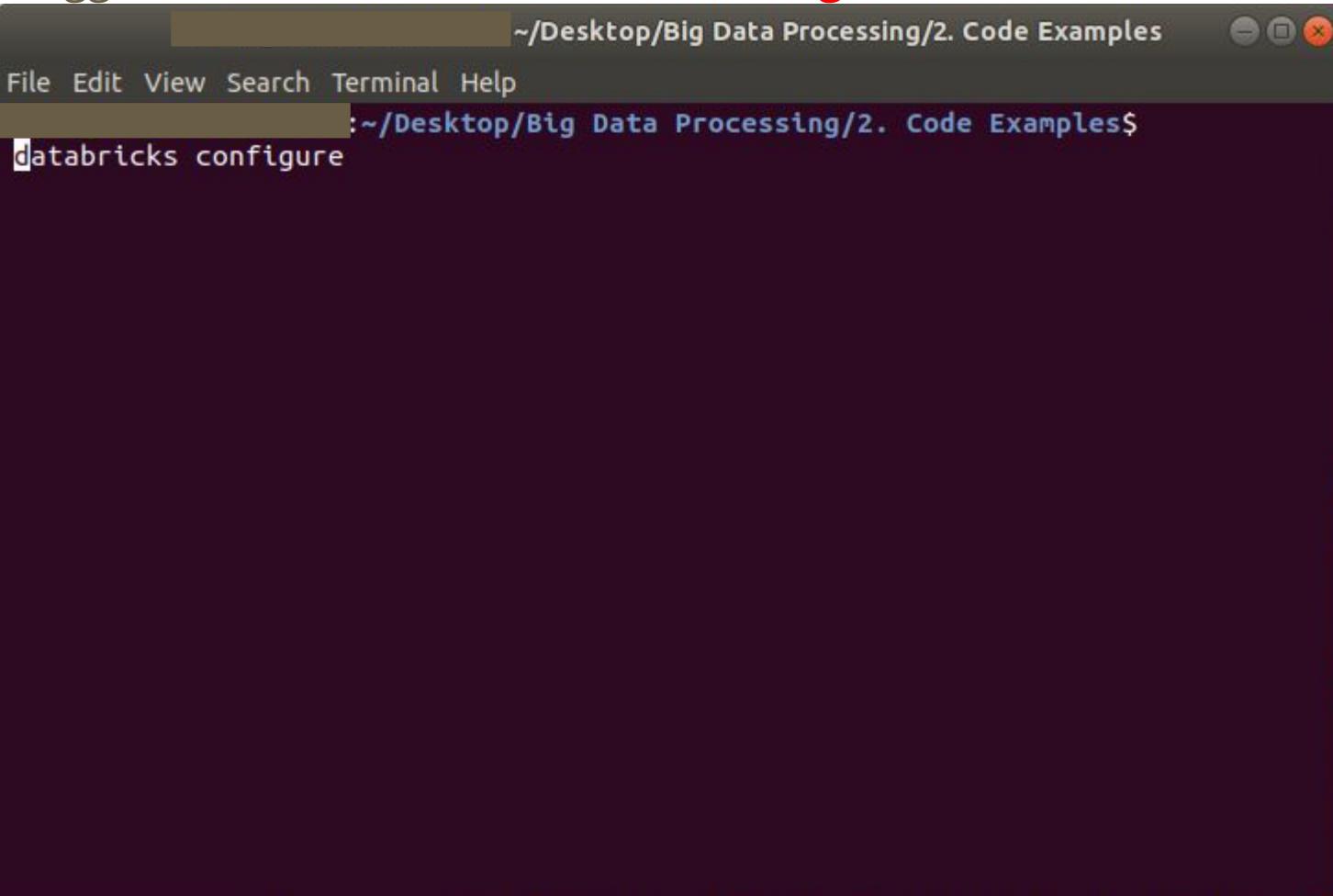
# Databricks: An Online Platform for Data Engineers

6. Databricks provides a Command Line Interface (CLI).  
Once installed, we must configure the Databricks CLI.

# Databricks: An Online Platform for Data Engineers

6. Databricks provides a Command Line Interface (CLI).

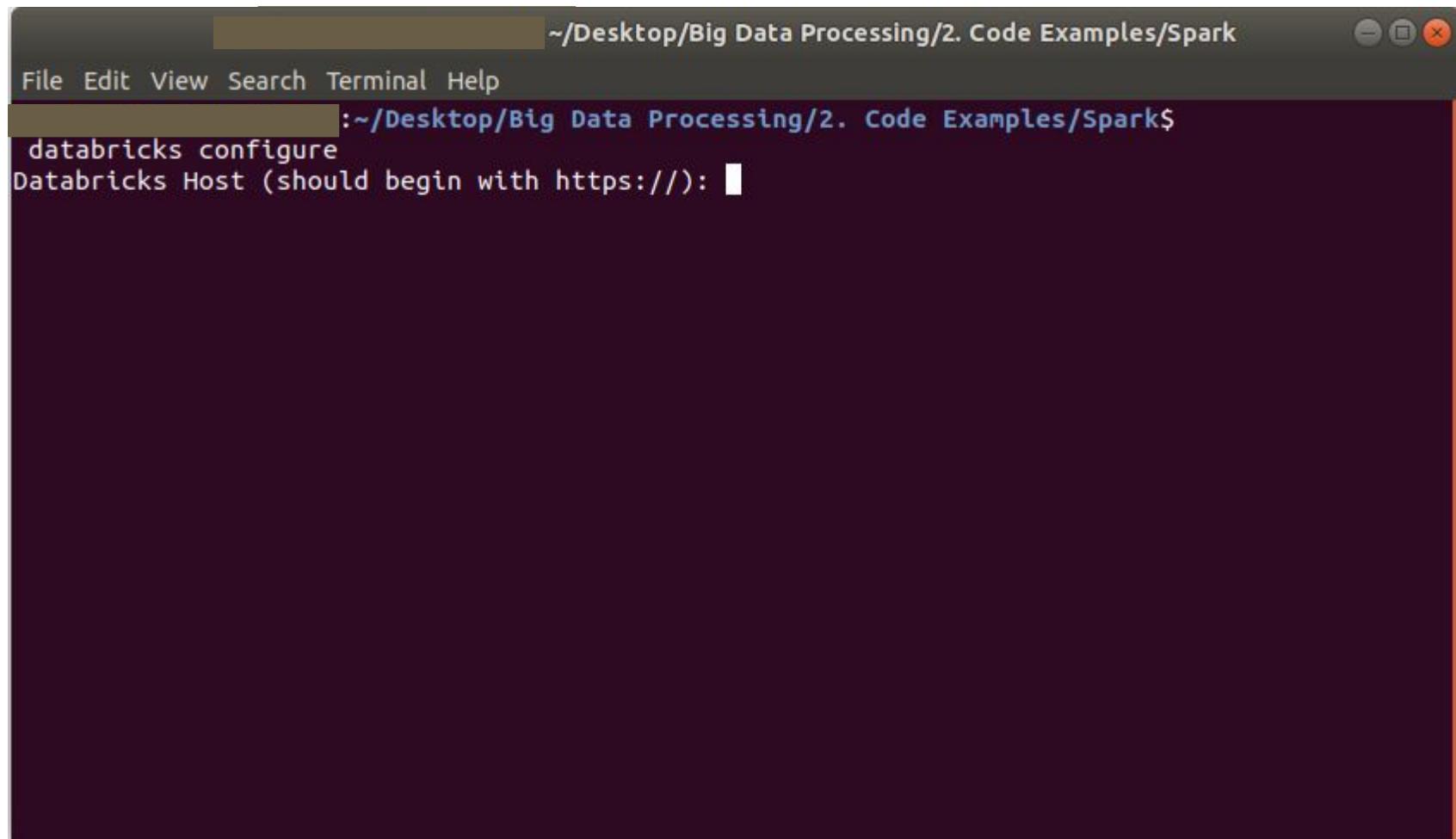
We trigger the command **databricks configure**



A screenshot of a terminal window titled '~/Desktop/Big Data Processing/2. Code Examples'. The window has a dark background and light-colored text. At the top, there is a menu bar with 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. Below the menu is a command prompt line: ':~/Desktop/Big Data Processing/2. Code Examples\$'. A cursor is visible at the beginning of the line, followed by the command 'databricks configure'. The rest of the terminal window is blank, showing a large white space.

# Databricks: An Online Platform for Data Engineers

6. Databricks provides a Command Line Interface (CLI).
  - a. We must indicate our databricks host.



A screenshot of a terminal window titled '~/Desktop/Big Data Processing/2. Code Examples/Spark'. The window has a dark theme with a light gray header bar. The title bar shows the path. The menu bar includes 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. The main terminal area shows the command:

```
~/Desktop/Big Data Processing/2. Code Examples/Spark$  
databricks configure  
Databricks Host (should begin with https://): █
```

The cursor is at the end of the host input field, indicated by a small black square.

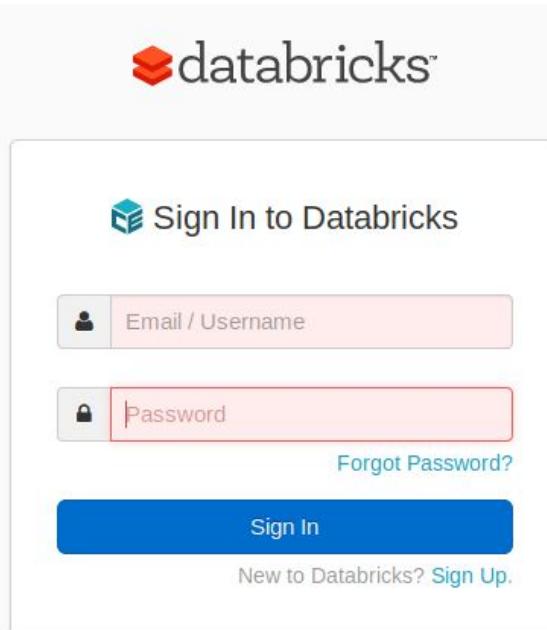
# Databricks: An Online Platform for Data Engineers

6. Databricks provides a Command Line Interface (CLI).

a. We must indicate our databricks host.

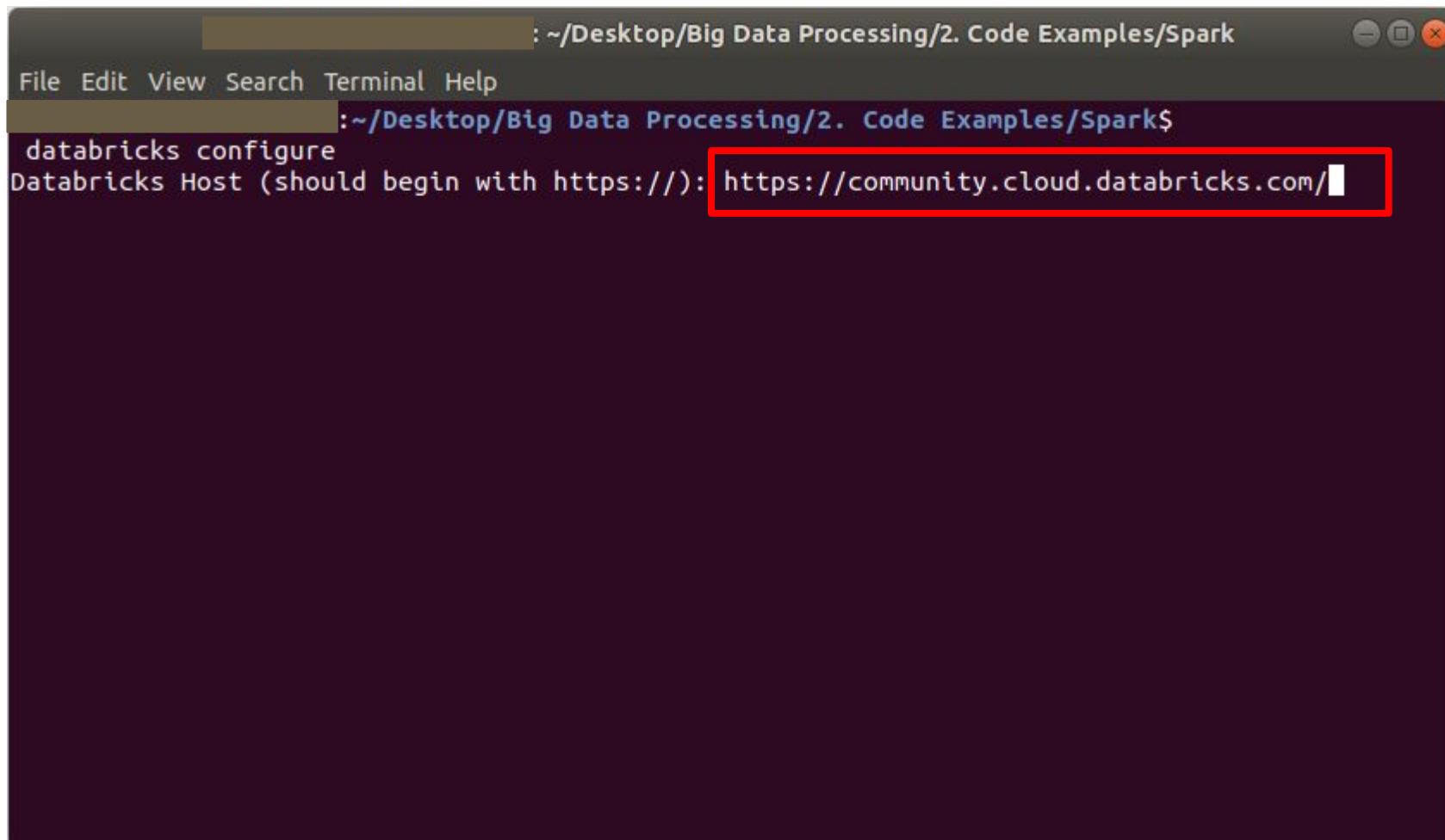
In our case, our host is the website for Databricks Community Edition:

<https://community.cloud.databricks.com>



# Databricks: An Online Platform for Data Engineers

6. Databricks provides a Command Line Interface (CLI).
  - a. We must indicate our databricks host.

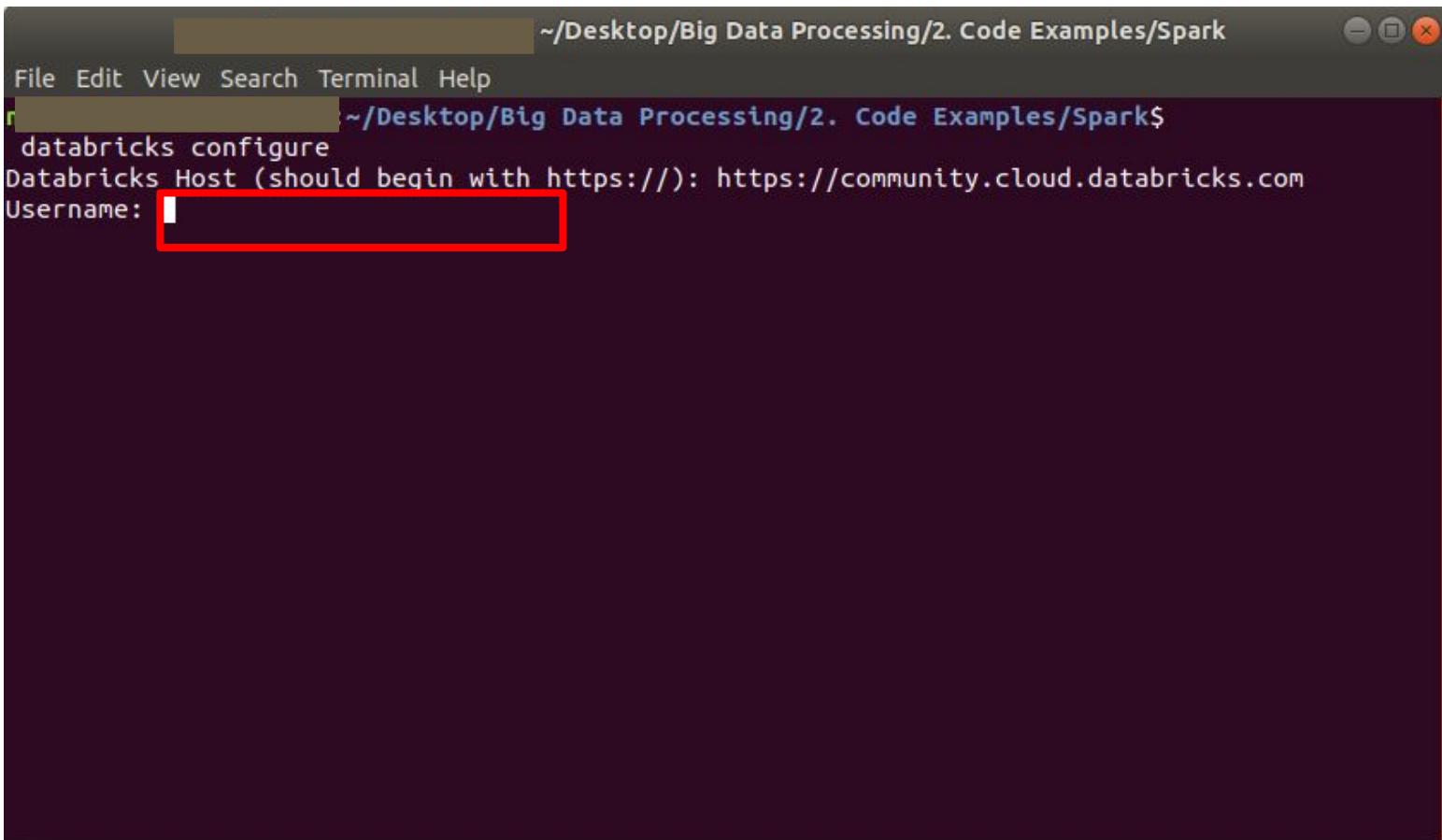


A screenshot of a terminal window titled 'Terminal' with the path '/Desktop/Big Data Processing/2. Code Examples/Spark'. The window has a dark background and light-colored text. A red box highlights the command 'databricks configure' and the prompt 'Databricks Host (should begin with https://):'. The user has typed 'https://community.cloud.databricks.com/' into the input field, which is also highlighted by a red box.

```
~/Desktop/Big Data Processing/2. Code Examples/Spark$ databricks configure
Databricks Host (should begin with https://): https://community.cloud.databricks.com/
```

# Databricks: An Online Platform for Data Engineers

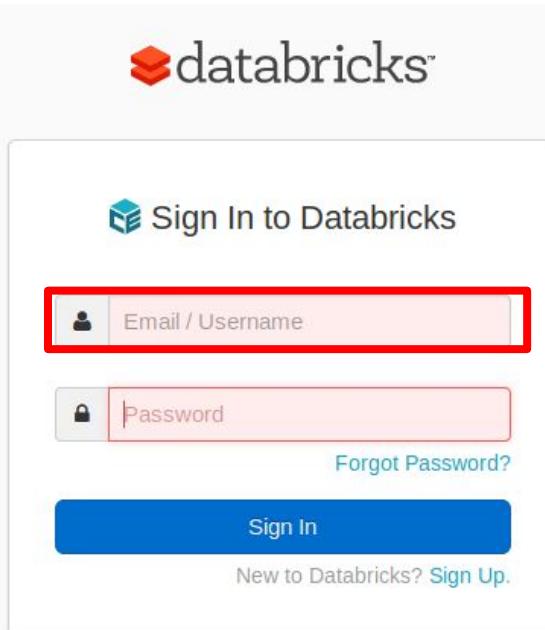
6. Databricks provides a Command Line Interface (CLI).
  - b. We must indicate our username (same as when we sign in at <https://community.cloud.databricks.com>).



The screenshot shows a terminal window with a dark background and light-colored text. The title bar reads: ~/Desktop/Big Data Processing/2. Code Examples/Spark. The menu bar includes: File, Edit, View, Search, Terminal, Help. The command line shows: databricks configure. Below the command, there is a prompt: Databricks Host (should begin with https://): https://community.cloud.databricks.com. A red rectangular box highlights the input field where the URL is typed. The terminal window has standard window controls (minimize, maximize, close) in the top right corner.

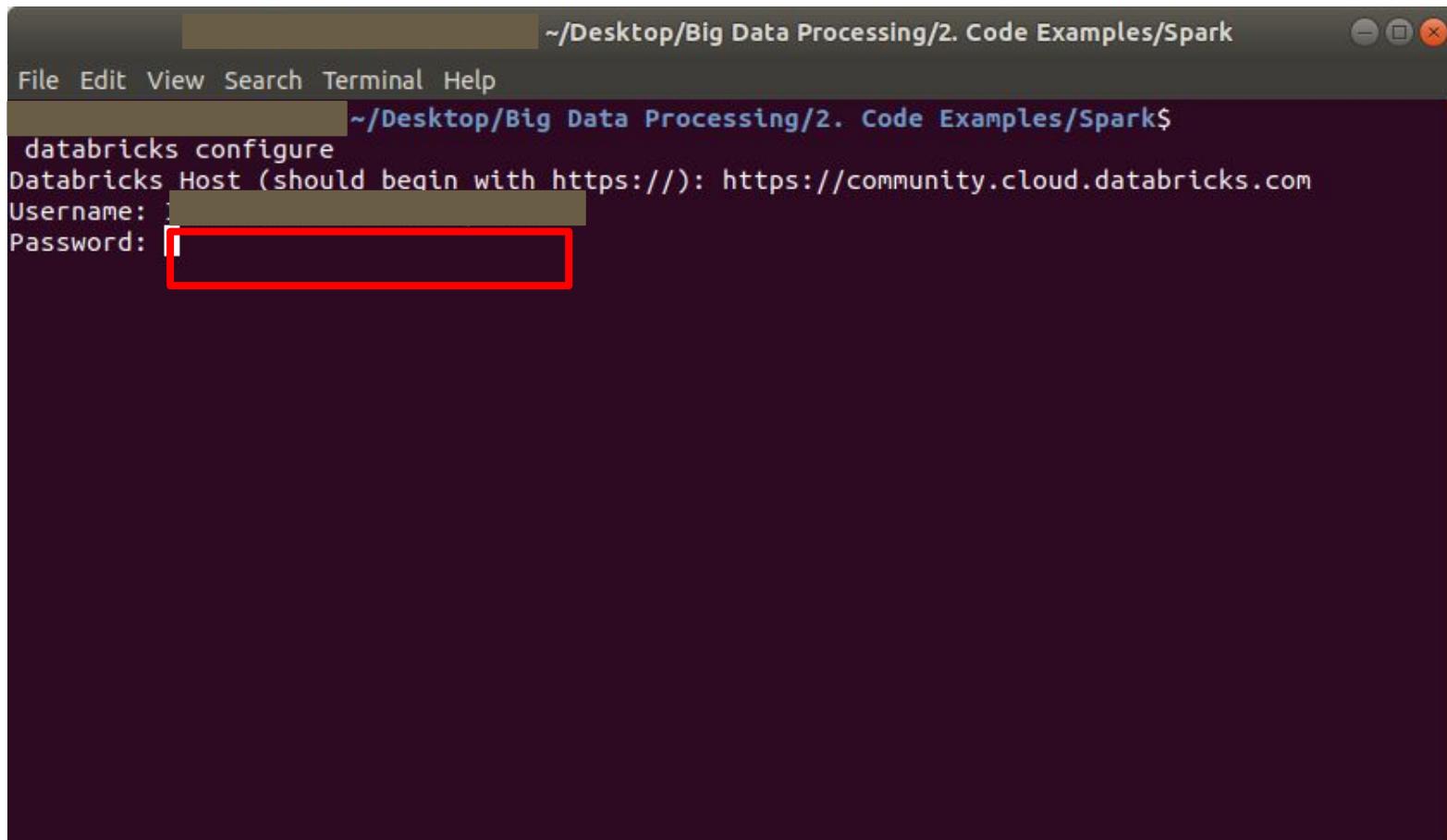
# Databricks: An Online Platform for Data Engineers

6. Databricks provides a Command Line Interface (CLI).
  - b. We must indicate our username (same as when we sign in at <https://community.cloud.databricks.com>).



# Databricks: An Online Platform for Data Engineers

6. Databricks provides a Command Line Interface (CLI).
  - c. We must indicate our password (same as when we sign in at <https://community.cloud.databricks.com>).

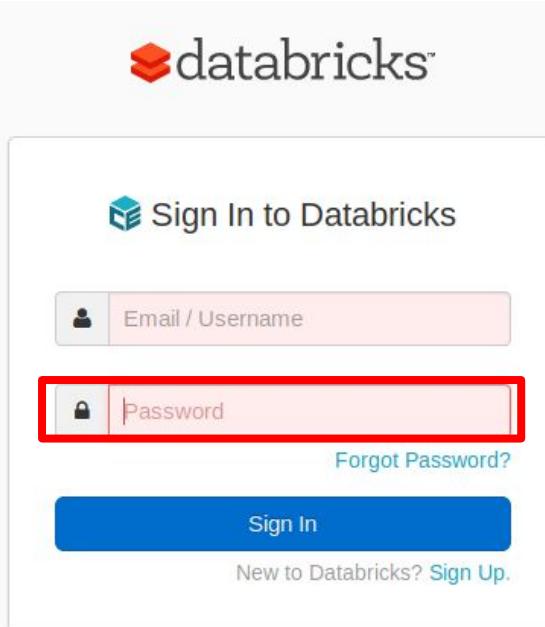


A screenshot of a terminal window titled '~/Desktop/Big Data Processing/2. Code Examples/Spark'. The window has a dark background and a light-colored title bar. The menu bar includes 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. The terminal prompt is ' ~/Desktop/Big Data Processing/2. Code Examples/Spark\$'. The command 'databricks configure' is entered, followed by prompts for 'Databricks Host (should begin with https://): https://community.cloud.databricks.com', 'Username:', and 'Password:'. The 'Password:' field is highlighted with a red rectangular border.

```
~/Desktop/Big Data Processing/2. Code Examples/Spark$  
databricks configure  
Databricks Host (should begin with https://): https://community.cloud.databricks.com  
Username:  
Password:
```

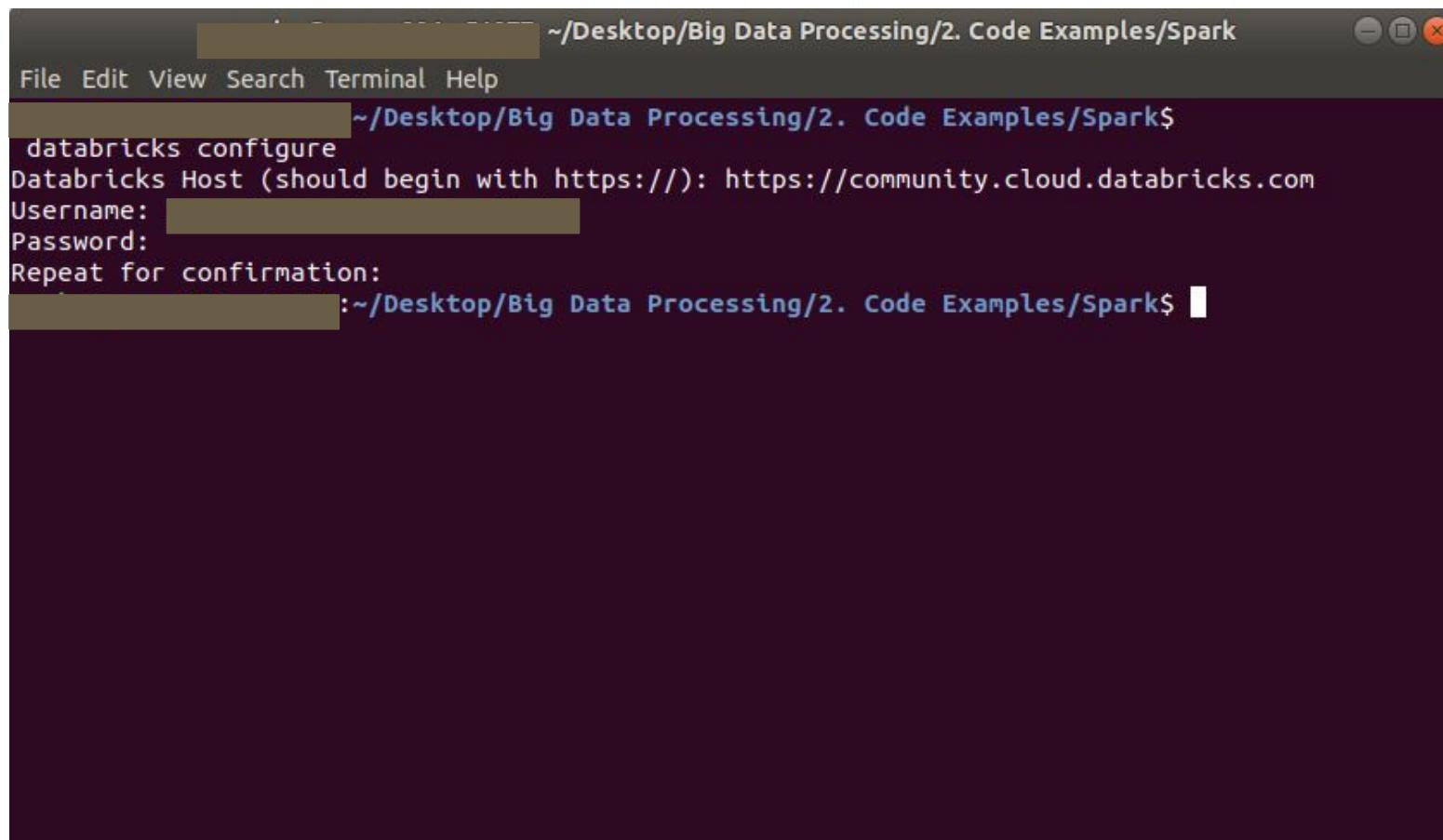
# Databricks: An Online Platform for Data Engineers

6. Databricks provides a Command Line Interface (CLI).
  - c. We must indicate our password (same as when we sign in at <https://community.cloud.databricks.com>).



# Databricks: An Online Platform for Data Engineers

6. Databricks provides a Command Line Interface (CLI).
  - c. We must indicate our password (same as when we sign in at <https://community.cloud.databricks.com>).



The screenshot shows a terminal window with a dark background and light-colored text. The title bar reads: ~/Desktop/Big Data Processing/2. Code Examples/Spark. The menu bar includes: File, Edit, View, Search, Terminal, Help. The command entered is: `databricks configure`. The response from the command line is:

```
~/Desktop/Big Data Processing/2. Code Examples/Spark$ databricks configure
Databricks Host (should begin with https://): https://community.cloud.databricks.com
Username: [REDACTED]
Password: [REDACTED]
Repeat for confirmation: [REDACTED]
~/Desktop/Big Data Processing/2. Code Examples/Spark$ █
```

# Databricks: An Online Platform for Data Engineers

*How to...*

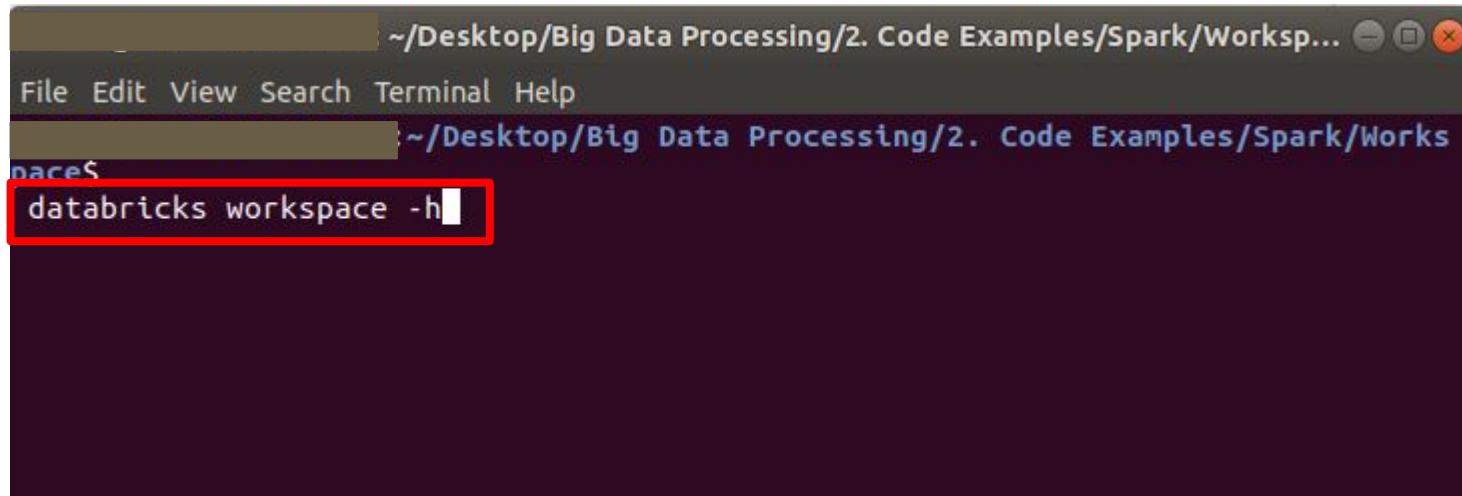
Upload an entire directory  
from our local machine  
to our Databricks Workspace

# Databricks: An Online Platform for Data Engineers

7. The Databricks CLI has some **workspace** specific commands:

# Databricks: An Online Platform for Data

7. The Databricks CLI has some **workspace** specific commands:



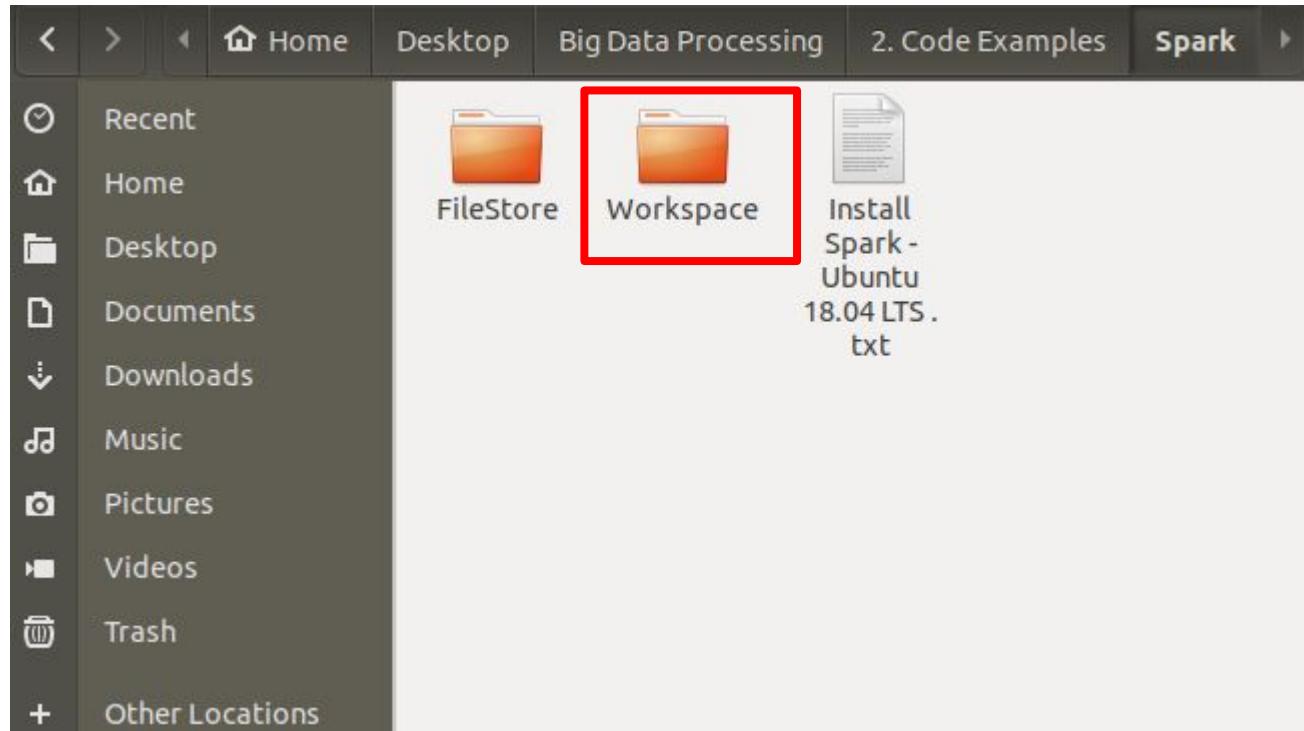
A screenshot of a terminal window titled ' ~/Desktop/Big Data Processing/2. Code Examples/Spark/Worksp...'. The window has a dark background and light-colored text. The menu bar includes 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. The current directory is shown in the title bar as ' ~/Desktop/Big Data Processing/2. Code Examples/Spark/Works...'. The command 'databricks workspace -h' is typed in the terminal, and it is highlighted with a red rectangle. The terminal is currently empty, showing only the command prompt.

```
~/Desktop/Big Data Processing/2. Code Examples/Spark/Worksp... ◻ ◻ ◻
File Edit View Search Terminal Help
:~/Desktop/Big Data Processing/2. Code Examples/Spark/Works
pace$ databricks workspace -h
Usage: databricks workspace [OPTIONS] COMMAND [ARGS]...
Utility to interact with the Databricks workspace. Workspace paths must be
absolute and be prefixed with `/`.
Options:
-v, --version    0.9.0
--debug          Debug Mode. Shows full stack trace on error.
--profile TEXT  CLI connection profile to use. The default profile is
                  "DEFAULT".
-h, --help        Show this message and exit.

Commands:
delete          Deletes objects from the Databricks workspace. rm and delete are
                  synonyms.
export          Exports a file from the Databricks workspace.
export_dir      Recursively exports a directory from the Databricks workspace.
import          Imports a file from local to the Databricks workspace.
import_dir      Recursively imports a directory to the Databricks workspace. import_dir
list            List objects in the Databricks workspace. ls and list are
                  synonyms.
ls              List objects in the Databricks Workspace. ls and list are
                  synonyms.
mkdirs          Make directories in the Databricks Workspace.
rm              Deletes objects from the Databricks workspace. rm and delete are
                  synonyms.
pace$
```

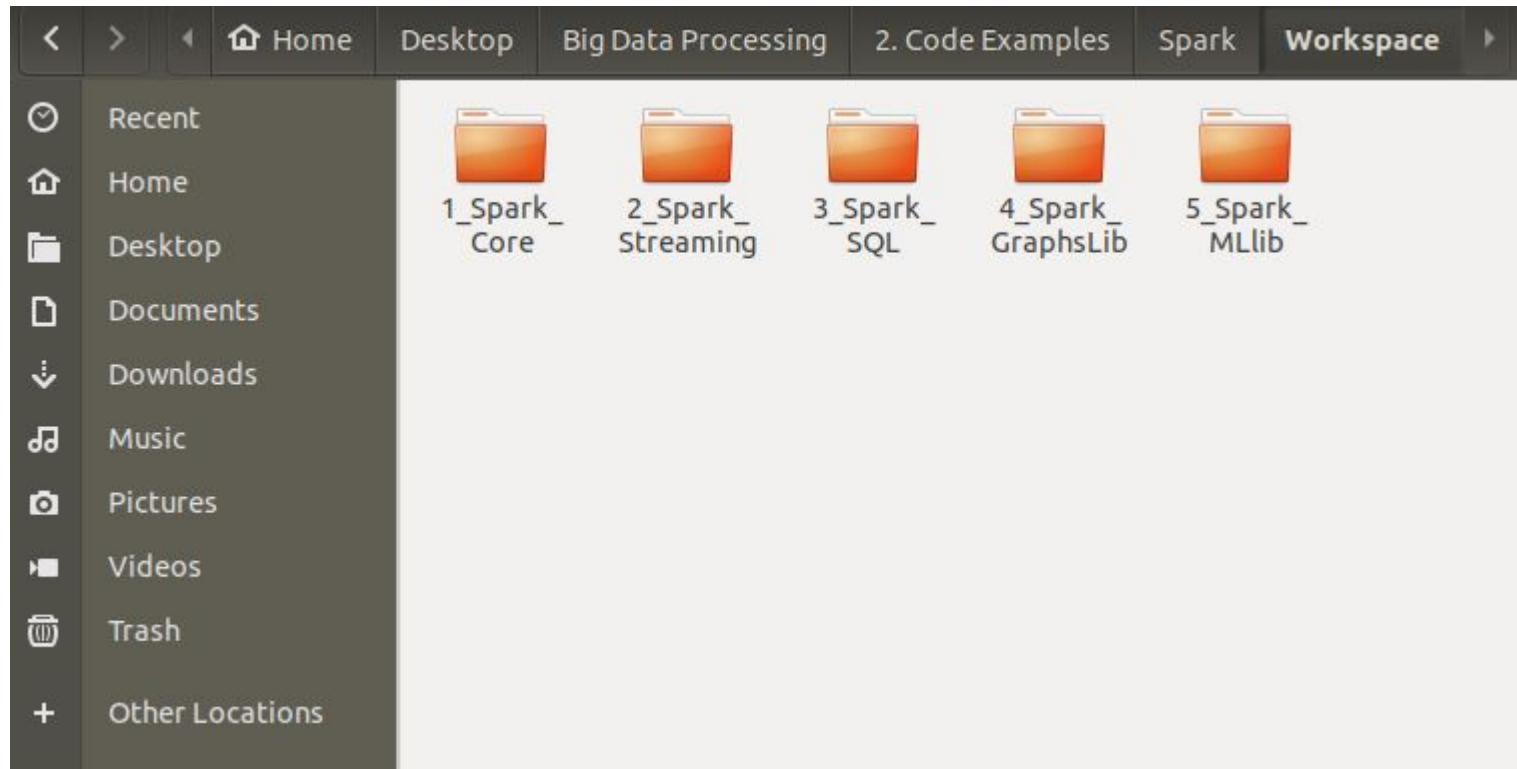
# Databricks: An Online Platform for Data Engineers

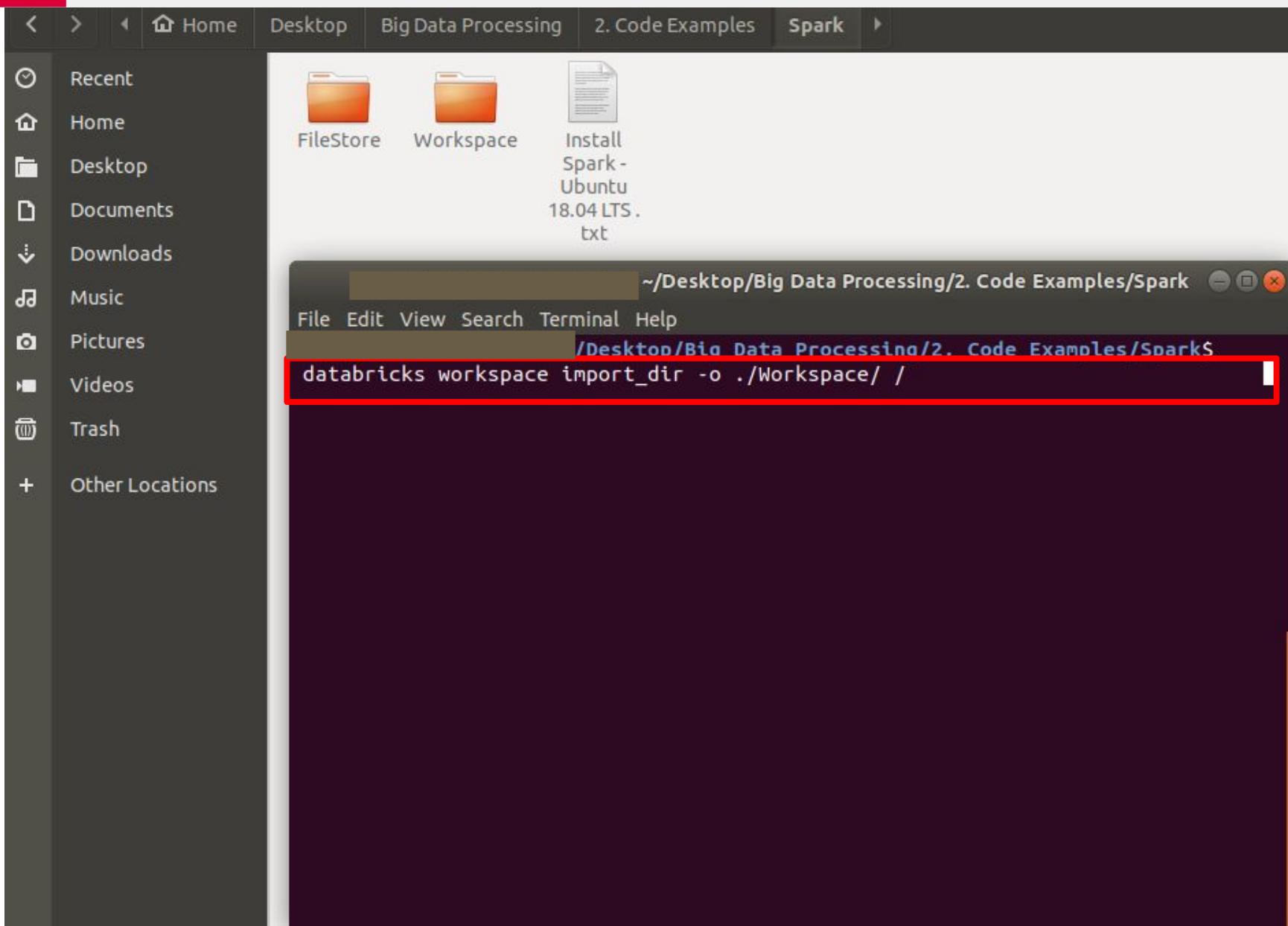
7. We can use the command **import\_dir** command to upload our set of Spark code examples (available in Canvas) to our Databricks Workspace.



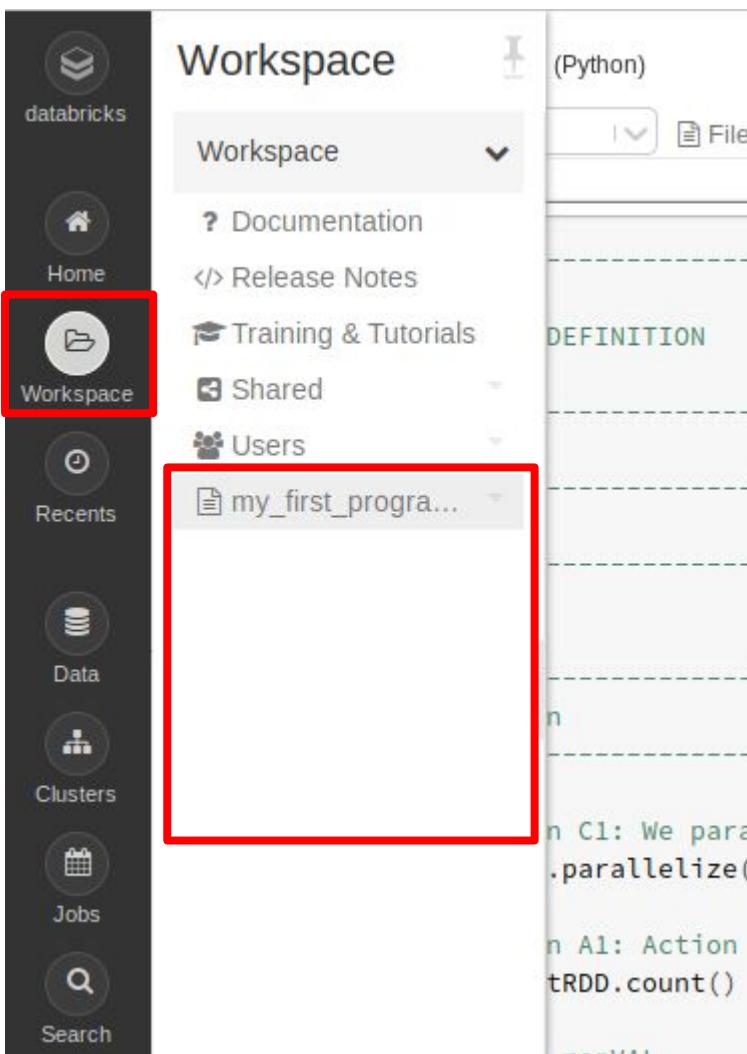
# Databricks: An Online Platform for Data Engineers

7. We can use the command **import\_dir** command to upload our set of Spark code examples (available in Canvas) to our Databricks Workspace.





# Databricks: An Online Platform for Data Engineering

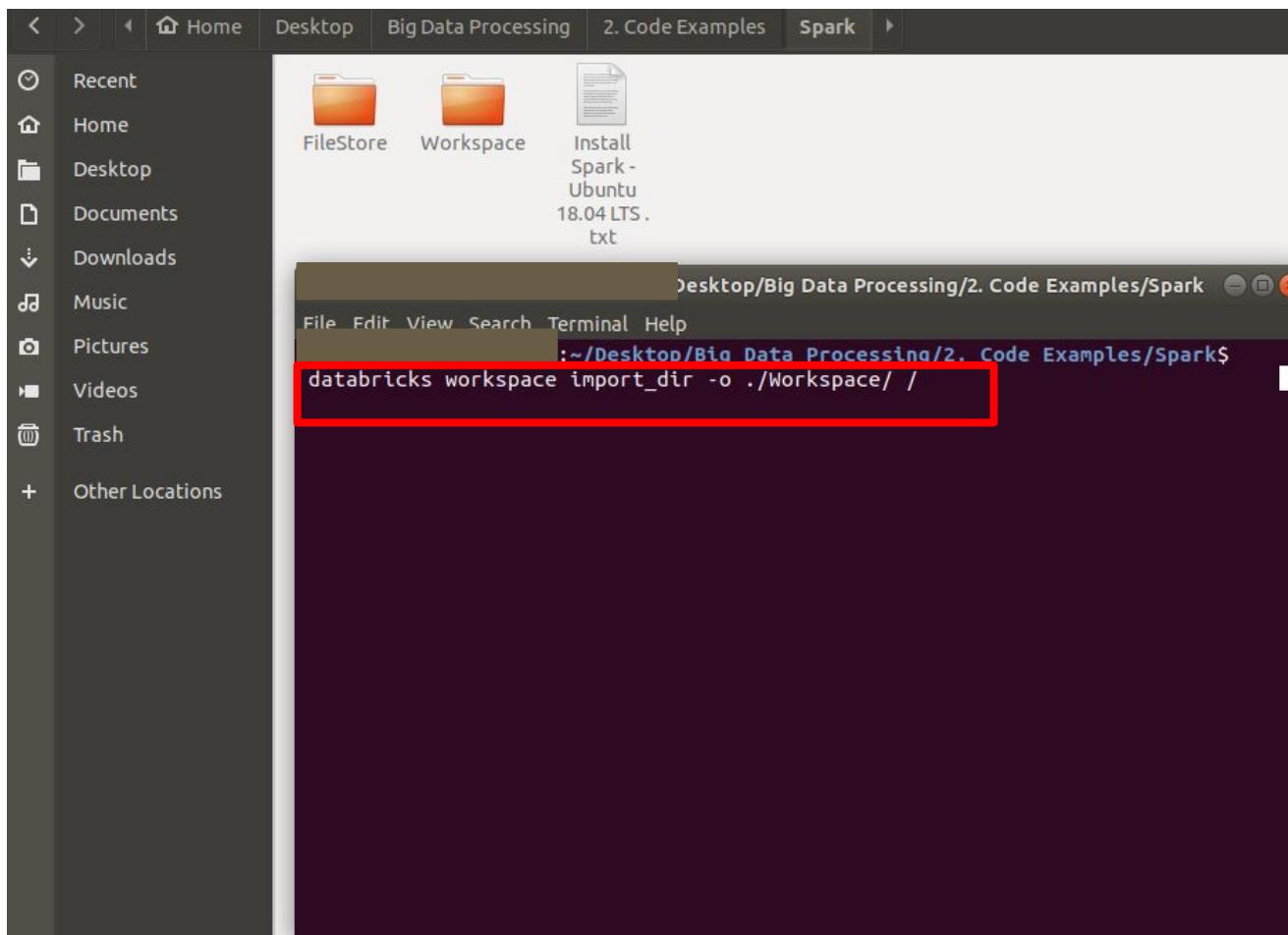


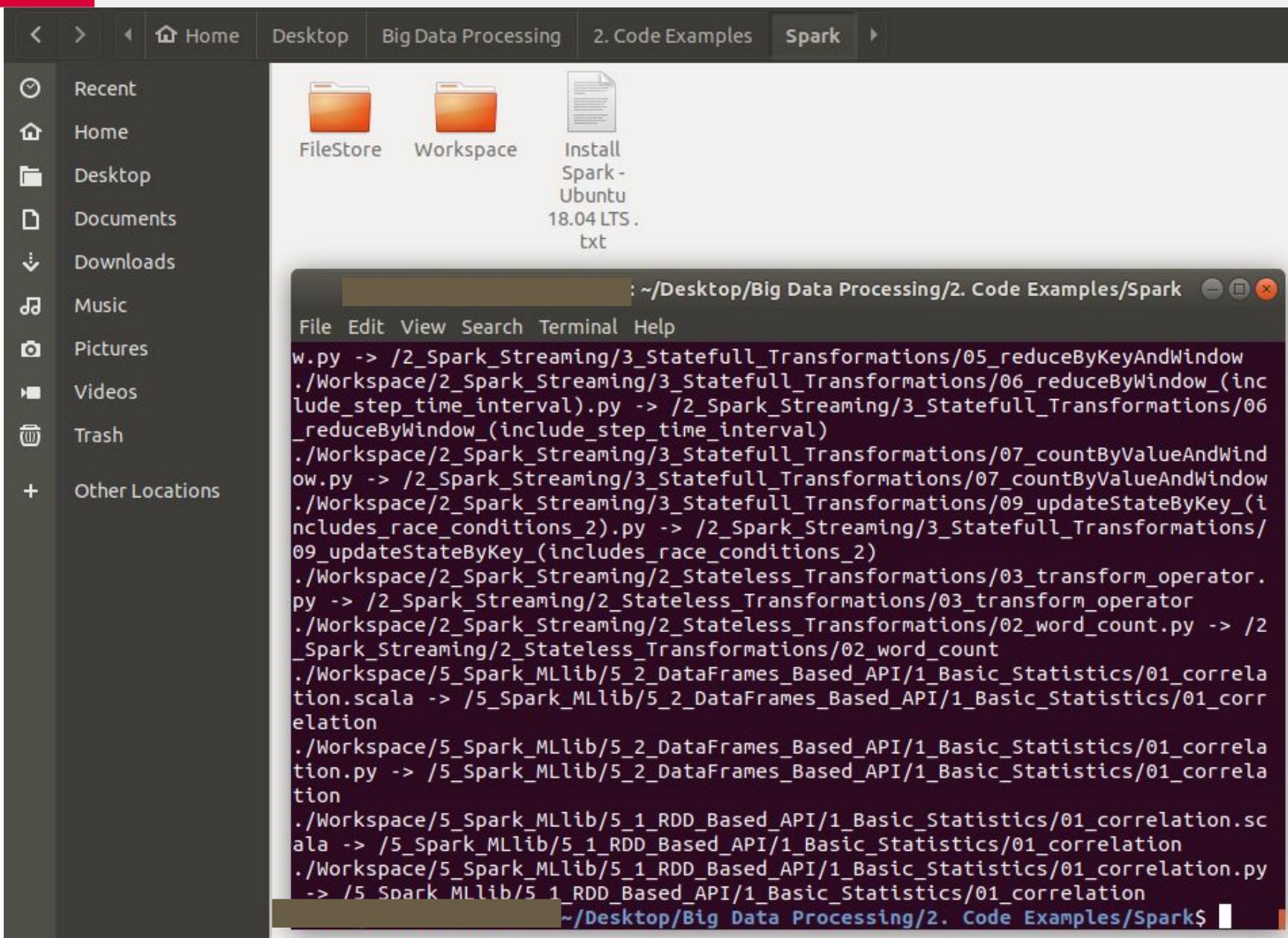
As we can see, before running the command, our **Databricks Workspace** only has the Python program we created before.

# Databricks: An Online Platform for Data Engineering

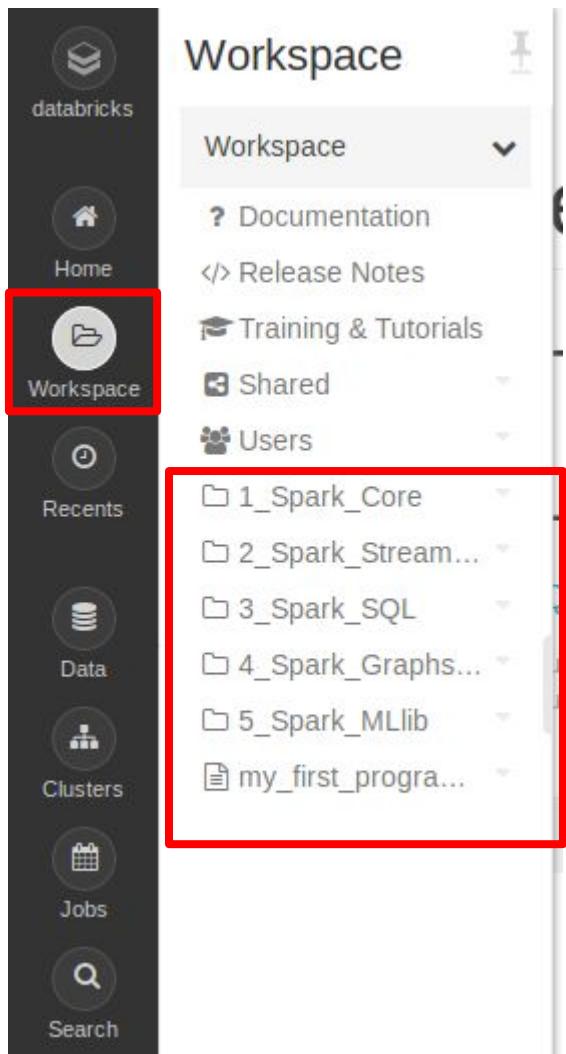
7. Running the command will take a couple of minutes.

```
> databricks workspace import_dir -o ./Workspace/ /
```





# Databricks: An Online Platform for Data Engineering



As we can see, after running the command, our **Databricks Workspace** contains our set of Spark code examples that we will use during the semester.

# Databricks: An Online Platform for Data Engineers

*How to...*  
Upload a Dataset  
from our local machine  
to the Databricks File System (DBFS).

# Databricks: An Online Platform for Data Engineers

8. We can see the files stored in the online Databricks File System (DBFS) by clicking in **Data**.



## Welcome to databricks™



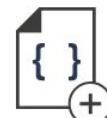
[Explore the Quickstart Tutorial](#)

Spin up a cluster, run queries on preloaded data, and display results in 5 minutes.



[Import & Explore Data](#)

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



[Create a Blank Notebook](#)

Create a notebook to start querying, visualizing, and modeling your data.

### Common Tasks

[New Notebook](#)

[Create Table](#)

[New Cluster](#)

### Recents

34\_job\_inspection\_5.scala

33\_job\_inspection\_4.scala

32\_job\_inspection\_3.scala

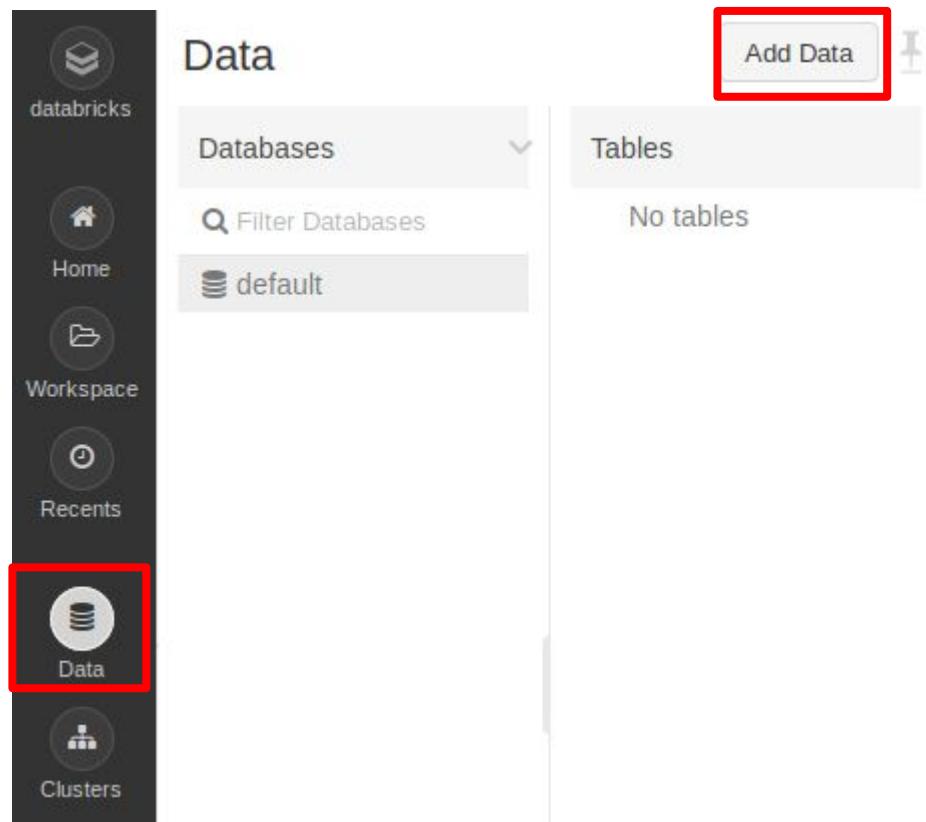
### What's new in v3.2

- Instance Pools
- Databricks Runtime 6.0 will drop Python 2 support

[View latest release notes](#)

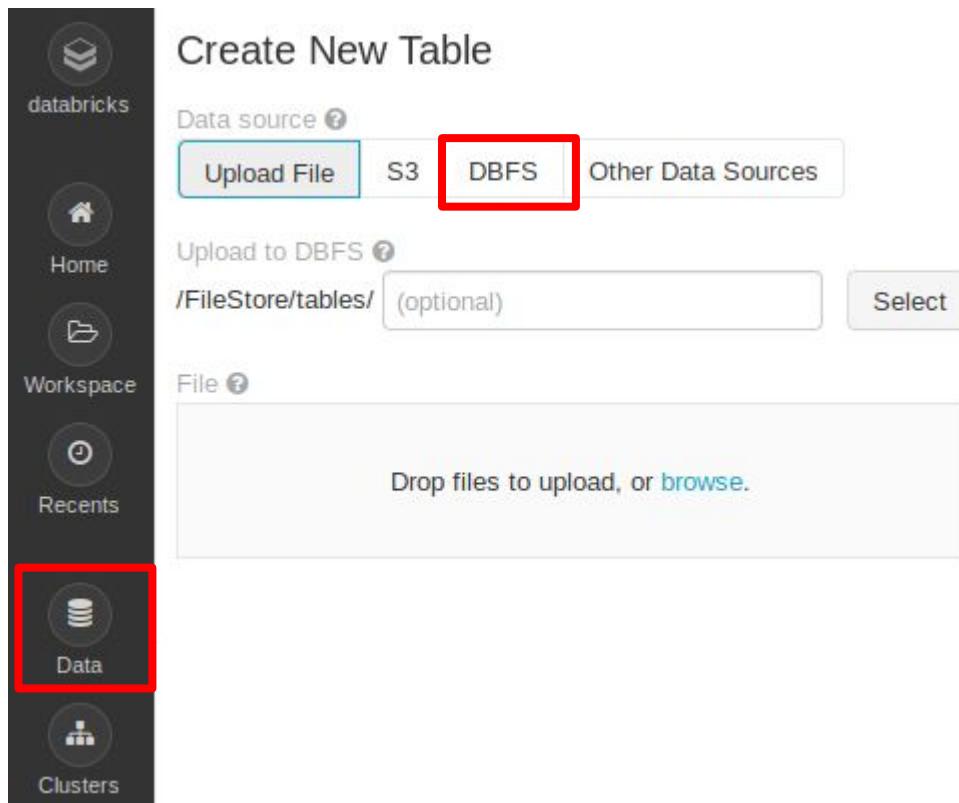
# Databricks: An Online Platform for Data Engineers

8. We can see the files stored in the online Databricks File System (DBFS) by clicking in **Data**.



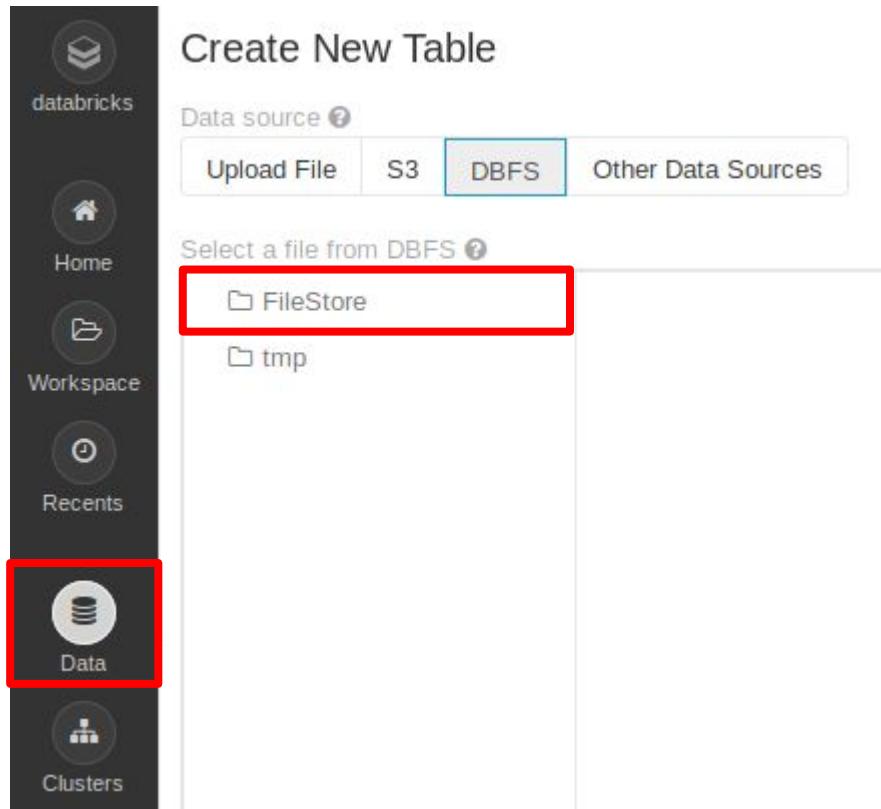
# Databricks: An Online Platform for Data Engineers

8. We can see the files stored in the online Databricks File System (DBFS) by clicking in **Data**.



# Databricks: An Online Platform for Data Engineers

8. We can see the files stored in the online Databricks File System (DBFS) by clicking in **Data**.



Any **Dataset** being used during the semester is to be placed at **dbfs:/FileStore/tables/**

# Databricks: An Online Platform for Data Engineers

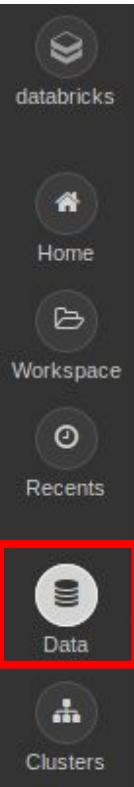
8. We can see the files stored in the online Databricks File System (DBFS) by clicking in **Data**.

The screenshot shows the Databricks Data interface. On the left is a sidebar with icons for Home, Workspace, Recents, Data (which is selected and highlighted with a red box), and Clusters. The main area is titled "Create New Table" and shows the "Data source" section with tabs for Upload File, S3, DBFS (which is selected and highlighted with a blue box), and Other Data Sources. Below this, a "Select a file from DBFS" section shows a tree view of the DBFS structure. The root directory "FileStore" is highlighted with a red box. Inside "FileStore", there are sub-directories "jars" and "tables", also highlighted with a red box. The "tmp" directory is shown below "FileStore".

Any **Dataset** being used during the semester is to be placed at  
**dbfs:/FileStore/tables/**

# Databricks: An Online Platform for Data Engineers

8. We can see the files stored in the online Databricks File System (DBFS) by clicking in **Data**.



Create New Table

Data source ?

Upload File S3 DBFS Other Data Sources

Select a file from DBFS ?

FileStore jars tables tmp

A screenshot of the Databricks interface showing the 'Create New Table' page. The 'DBFS' tab is selected under 'Data source'. In the 'Select a file from DBFS' section, the 'FileStore' folder is expanded, showing its contents: 'jars', 'tables', and 'tmp'. The 'FileStore' folder and its sub-folders 'jars' and 'tables' are all highlighted with red boxes. A large red box also surrounds the entire 'FileStore' section.

Any **Dataset** being used during the semester is to be placed at **dbfs:/FileStore/tables/**

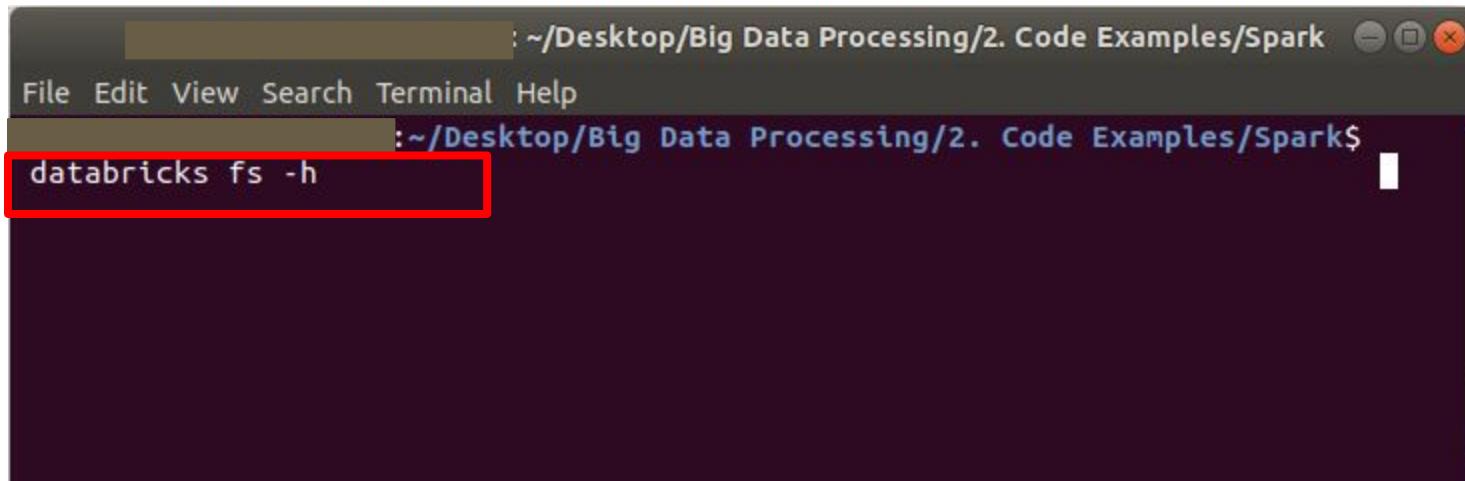
As we can see by the moment there is no Dataset store at DBFS.

# Databricks: An Online Platform for Data Engineers

8. The CLI has some **Databricks File System (DBFS)** specific commands:

# Databricks: An Online Platform for Data Engineers

8. The CLI has some **Databricks File System (DBFS)** specific commands:



A screenshot of a terminal window titled 'Terminal'. The window shows the path '/Desktop/Big Data Processing/2. Code Examples/Spark'. The command 'databricks fs -h' is entered at the prompt. The entire command line is highlighted with a red rectangular box.

```
:~/Desktop/Big Data Processing/2. Code Examples/Spark$ databricks fs -h
```

```
~/Desktop/Big Data Processing/2. Code Examples/Spark
```



```
File Edit View Search Terminal Help
```

```
~/Desktop/Big Data Processing/2. Code Examples/Spark$
```

```
databricks fs -h
```

```
Usage: databricks fs [OPTIONS] COMMAND [ARGS]...
```

```
Utility to interact with DBFS.
```

```
DBFS paths are all prefixed with dbfs:. Local paths can be absolute or local.
```

```
Options:
```

```
-v, --version    0.9.0
```

```
--debug         Debug Mode. Shows full stack trace on error.
```

```
--profile TEXT CLI connection profile to use. The default profile is "DEFAULT".
```

```
-h, --help       Show this message and exit.
```

```
Commands:
```

```
cat      Show the contents of a file.
```

```
configure Configures host and authentication info for the CLI.
```

```
cp       Copy files to and from DBFS.
```

```
ls       List files in DBFS.
```

```
mkdirs   Make directories in DBFS.
```

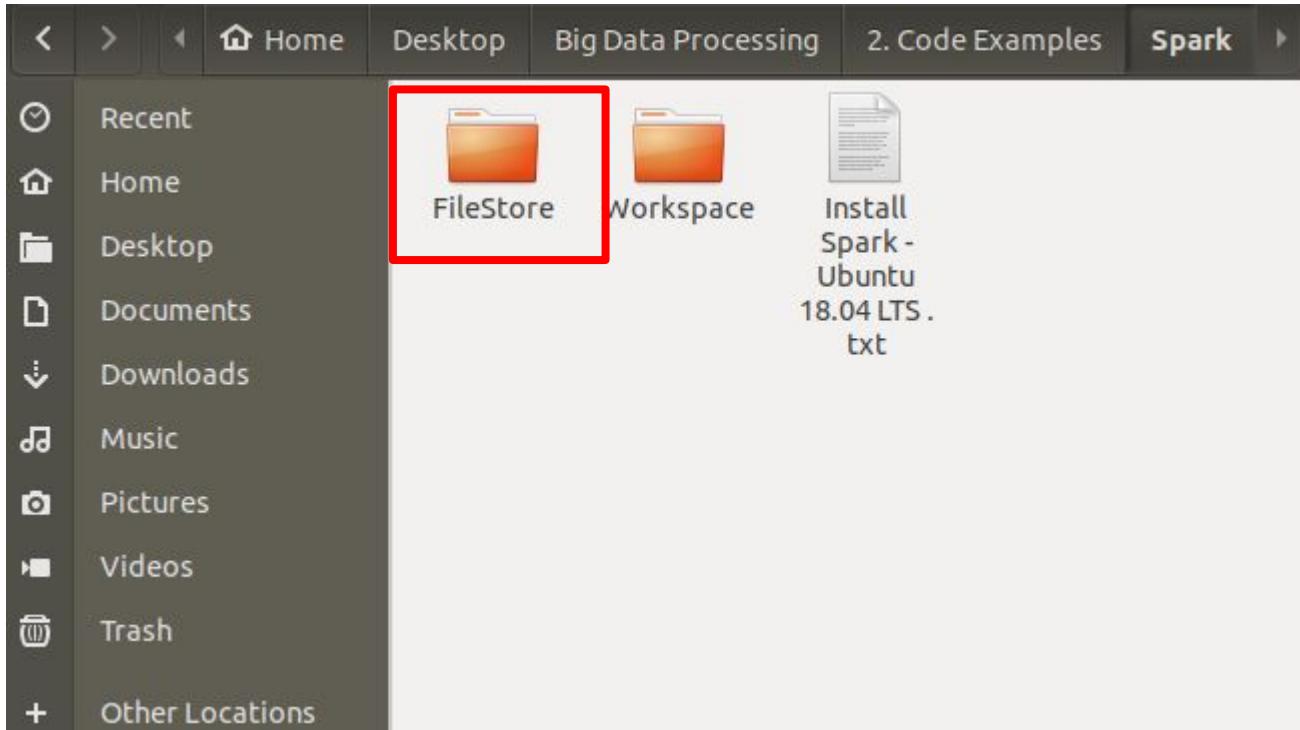
```
mv       Moves a file between two DBFS paths.
```

```
rm       Remove files from dbfs.
```

```
~/Desktop/Big Data Processing/2. Code Examples/Spark$
```

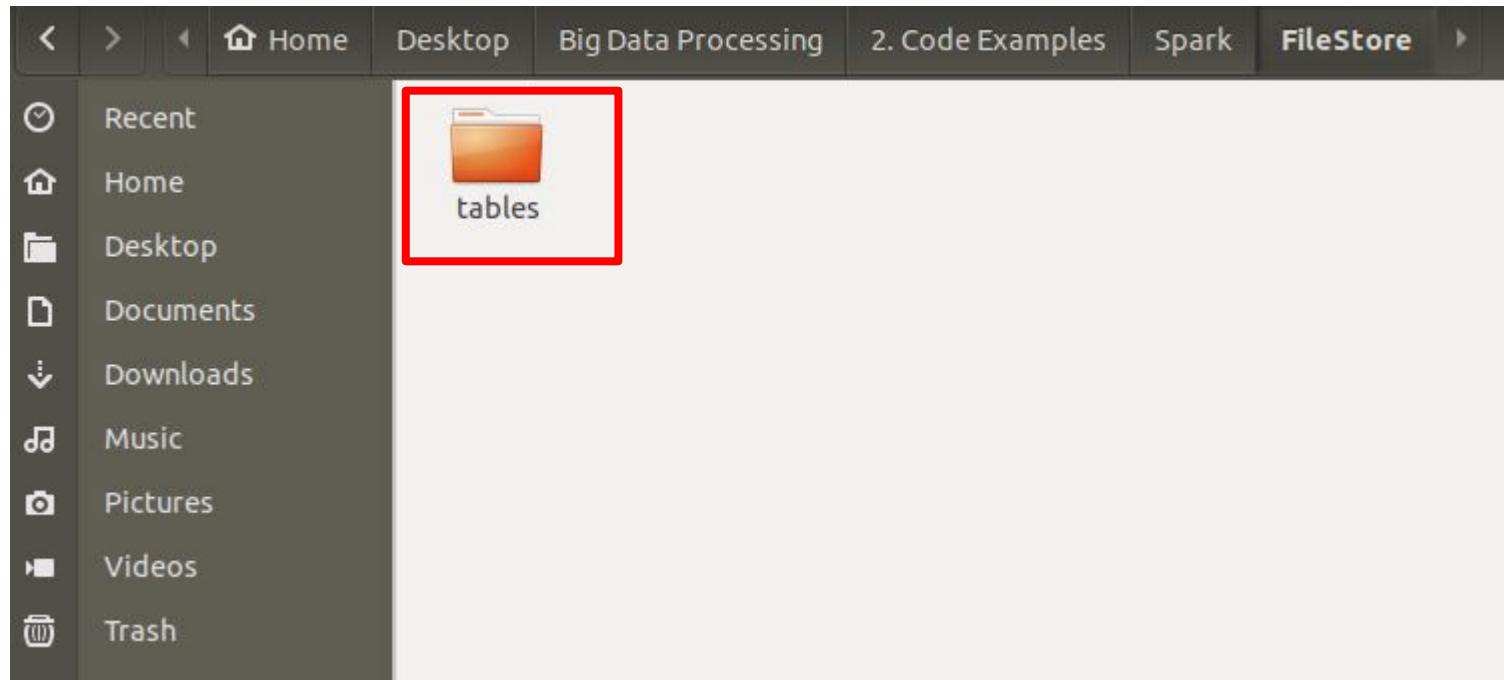
# Databricks: An Online Platform for Data Engineers

8. We can use the command **cp** to upload our set of Datasets to DBFS.



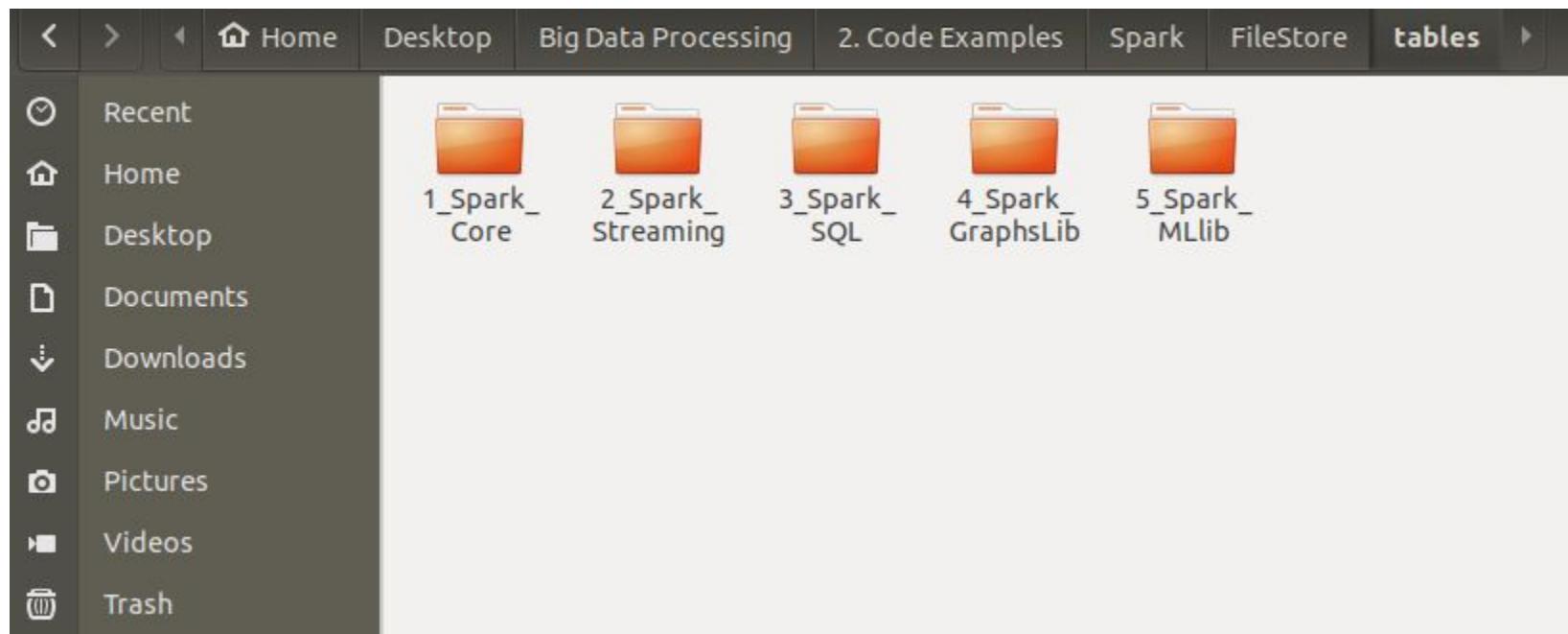
# Databricks: An Online Platform for Data Engineers

8. We can use the command **cp** to upload our set of Datasets to DBFS.



# Databricks: An Online Platform for Data Engineers

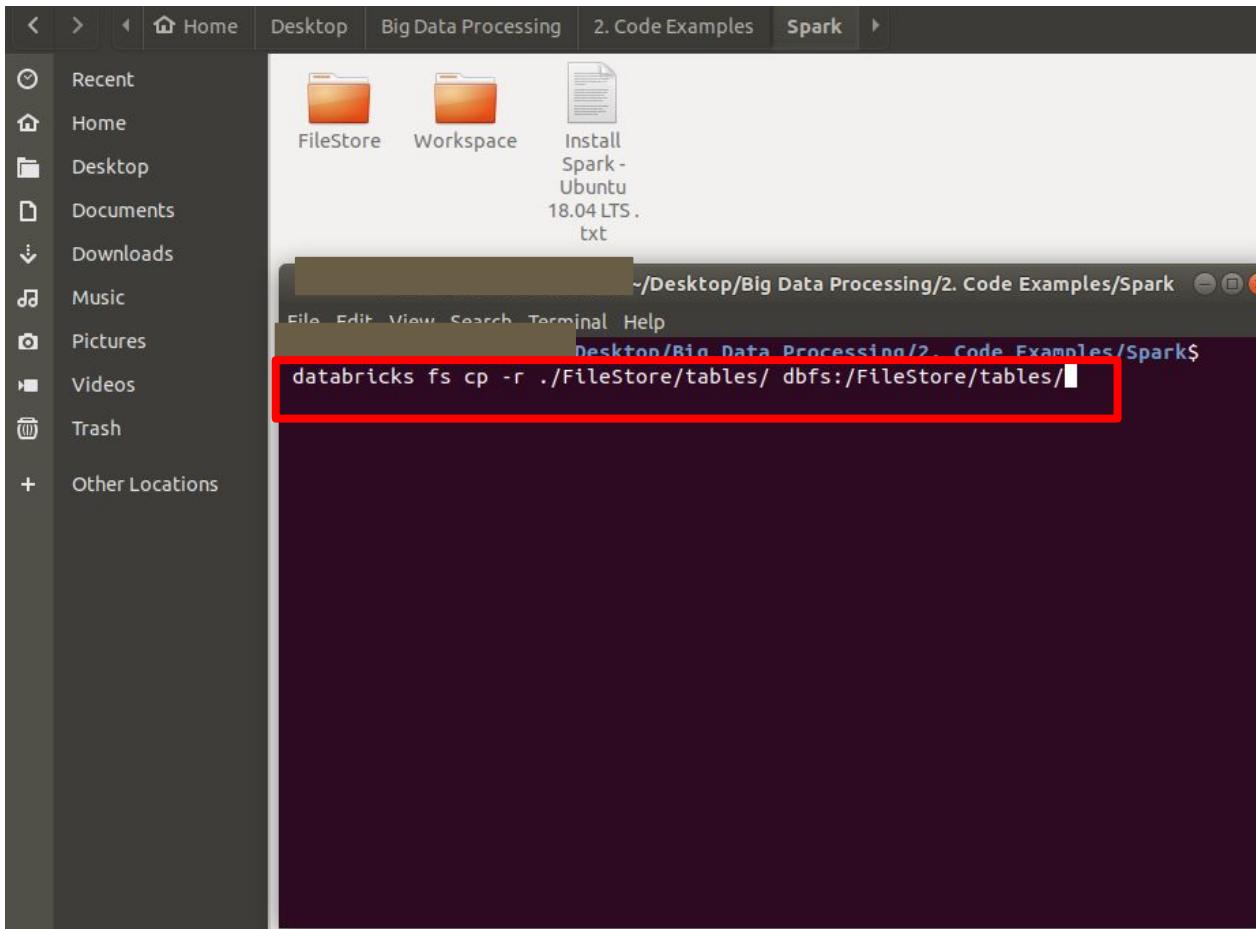
8. We can use the command **cp** to upload our set of Datasets to DBFS.



# Databricks: An Online Platform for Data Engin

8. Running the command will take a couple of minutes.

```
> databricks fs cp -r ./FileStore/tables dbfs:/FileStore/tables/
```



# Databricks: An Online Platform for Data Engin

8. Running the command will take a couple of minutes.

```
> databricks fs cp -r ./FileStore/tables dbfs:/FileStore/tables/
```

The screenshot shows a desktop environment with a file manager window open on the left and a terminal window open on the right. The file manager window displays a sidebar with links to Recent, Home, Desktop, Documents, Downloads, Music, Pictures, Videos, and Trash. It also shows icons for FileStore and Workspace, and a file named 'Install Spark - Ubuntu 18.04 LTS.txt'. The terminal window is titled '~/.Desktop/Big Data Processing/2. Code Examples/Spark' and contains the output of a 'databricks fs cp' command. The command lists multiple files being copied from the local 'FileStore' to the Databricks 'dbfs' storage. The files listed are: merge\_solutions.py, my\_tiny\_file\_03.txt, my\_tiny\_file\_02.txt, my\_tiny\_file\_05.txt, my\_tiny\_file\_04.txt, my\_tiny\_file\_01.txt, my\_tiny\_file\_06.txt, pearson\_3\_vars\_dataset.csv, pearson\_2\_vars\_dataset.csv, and spearman\_2\_vars\_dataset.csv.

```
File Edit View Search Terminal Help
./FileStore/tables/2_Spark_Streaming/merge_solutions.py -> dbfs:/FileStore/tables/2_Spark_Streaming/merge_solutions.py
./FileStore/tables/2_Spark_Streaming/my_dataset/my_tiny_file_03.txt -> dbfs:/FileStore/tables/2_Spark_Streaming/my_dataset/my_tiny_file_03.txt
./FileStore/tables/2_Spark_Streaming/my_dataset/my_tiny_file_02.txt -> dbfs:/FileStore/tables/2_Spark_Streaming/my_dataset/my_tiny_file_02.txt
./FileStore/tables/2_Spark_Streaming/my_dataset/my_tiny_file_05.txt -> dbfs:/FileStore/tables/2_Spark_Streaming/my_dataset/my_tiny_file_05.txt
./FileStore/tables/2_Spark_Streaming/my_dataset/my_tiny_file_04.txt -> dbfs:/FileStore/tables/2_Spark_Streaming/my_dataset/my_tiny_file_04.txt
./FileStore/tables/2_Spark_Streaming/my_dataset/my_tiny_file_01.txt -> dbfs:/FileStore/tables/2_Spark_Streaming/my_dataset/my_tiny_file_01.txt
./FileStore/tables/2_Spark_Streaming/my_dataset/my_tiny_file_06.txt -> dbfs:/FileStore/tables/2_Spark_Streaming/my_dataset/my_tiny_file_06.txt
./FileStore/tables/5_Spark_MLLib/1_Basic_Statistics/pearson_3_vars_dataset.csv -> dbfs:/FileStore/tables/5_Spark_MLLib/1_Basic_Statistics/pearson_3_vars_dataset.csv
./FileStore/tables/5_Spark_MLLib/1_Basic_Statistics/pearson_2_vars_dataset.csv -> dbfs:/FileStore/tables/5_Spark_MLLib/1_Basic_Statistics/pearson_2_vars_dataset.csv
./FileStore/tables/5_Spark_MLLib/1_Basic_Statistics/spearman_2_vars_dataset.csv -> dbfs:/FileStore/tables/5_Spark_MLLib/1_Basic_Statistics/spearman_2_vars_dataset.csv
```

# Databricks: An Online Platform for Data Engineering

The screenshot shows the Databricks interface for creating a new table. On the left, there's a sidebar with icons for Home, Workspace, Recents, Data (which is highlighted with a red box), and Clusters. The main area is titled 'Create New Table' and has a 'Data source' section with tabs for Upload File, S3, DBFS (which is selected and highlighted with a blue box), and Other Data Sources. Below this, it says 'Select a file from DBFS' and shows a tree view of the DBFS structure. The 'FileStore' folder is expanded, showing 'jars' and 'tables'. The 'tables' folder is also expanded, showing sub-folders named '1\_Spark\_Core', '2\_Spark\_Streaming', '3\_Spark\_SQL', '4\_Spark\_GraphsLib', and '5\_Spark\_MLlib'. Red boxes highlight the 'DBFS' tab, the 'FileStore' folder, and the 'tables' folder.

As we can see, after running the command, our **Datasets** are now stored at `dbfs:/FileStore/tables/`

# Databricks: An Online Platform for Data Engineers

*How to...*

Run a program / Spark Application  
reading in a Dataset from DBFS  
and possibly storing the results at DBFS.

# Databricks: An Online Platform for Data Engineering

- As we can see, Spark Core contains **my\_dataset** of 4 text files:  
comedies.txt    histories.txt    poems.txt    tragedies.txt

The screenshot shows the Databricks Data interface. On the left, there's a sidebar with icons for Home, Workspace, Recents, Data (which is selected and highlighted with a red box), and Clusters. The main area is titled "Create New Table" and shows a "Data source" section with tabs for Upload File, S3, DBFS (which is selected and highlighted with a blue border), and Other Data Sources. Below this, a "Select a file from DBFS" section shows a tree view of files under "DBFS". The tree includes "jars" and "tables" (which is selected and highlighted with a grey background). Under "tables", there are five items: "1\_Spark\_Core", "2\_Spark\_Streaming", "3\_Spark\_SQL", "4\_Spark\_GraphsLib", and "5\_Spark\_MLib". To the right of these, there are two more sections: "my\_result\_merge.py" and "my\_result". Under "my\_result", there are four files: "comedies.txt", "histories.txt", "poems.txt", and "tragedies.txt". The "1\_Spark\_Core", "my\_dataset", and the entire "my\_result" section are all highlighted with red boxes.

Create New Table

Data source ?

Upload File S3 DBFS Other Data Sources

Select a file from DBFS ?

jars

tables

1\_Spark\_Core

2\_Spark\_Streaming

3\_Spark\_SQL

4\_Spark\_GraphsLib

5\_Spark\_MLib

my\_result\_merge.py

my\_dataset

my\_result

comedies.txt

histories.txt

poems.txt

tragedies.txt

# Databricks: An Online Platform for Data Engineering

- As we can see, Spark Core contains the folder **my\_result**, being empty.

Create New Table

Data source

Upload File S3 DBFS Other Data Sources

Select a file from DBFS

jars

tables

1\_Spark\_Core

2\_Spark\_Streaming

3\_Spark\_SQL

4\_Spark\_GraphsLib

5\_Spark\_MLLib

my\_result\_merge.py

my\_dataset

my\_result

Data

Clusters

# Databricks: An Online Platform for Data Engineering

- Without explaining the program itself, our code example:

[Workspace/1\\_Spark\\_Core/6\\_Text\\_File\\_Examples/27\\_word\\_count.py](#)

The screenshot shows the Databricks workspace interface. On the left is a dark sidebar with icons for Home, Workspace (which is selected and highlighted with a red box), Recents, Data, Clusters, and Jobs. The main workspace area has two dropdown menus. The top-left dropdown is set to '1\_Spark\_Core' and contains items: 1\_Creation\_Operations, 2\_RDD\_Basic\_Transformations, 3\_RDD\_Basic\_Actions, 4\_RDD\_Key-Value\_Transforma..., 5\_RDD\_Key-Value\_Actions, 6\_Text\_File\_Examples (which is also highlighted with a red box), and 7\_Advance\_Features. The top-right dropdown is set to '6\_Text\_File\_Examples' and contains items: 25\_average\_length\_of\_words, 26\_average\_and\_percentage, and 27\_word\_count (which is also highlighted with a red box). Navigation arrows are visible at the bottom of the workspace area.

# Databricks: An Online Platform for Data Engin

- Without explaining the program itself, our code example

[Workspace/1\\_Spark\\_Core/6\\_Text\\_File\\_Examples/27\\_word\\_count.py](#)

- Reads the **Dataset** from  
**dbfs:/FileStore/1\_Spark\_Core/my\_dataset**

# Databricks: An Online Platform for Data Engin

- Without explaining the program itself, our code example

[Workspace/1\\_Spark\\_Core/6\\_Text\\_File\\_Examples/27\\_word\\_count.py](#)

- Reads the **Dataset** from  
**dbfs:/FileStore/1\_Spark\_Core/my\_dataset**
- Computes the word count of the words appearing in the files of  
my\_dataset: comedies.txt, histories.txt, poems.txt, tragedies.txt

# Databricks: An Online Platform for Data Engin

- Without explaining the program itself, our code example

[Workspace/1\\_Spark\\_Core/6\\_Text\\_File\\_Examples/27\\_word\\_count.py](#)

- Reads the **Dataset** from  
**dbfs:/FileStore/1\_Spark\_Core/my\_dataset**
- Computes the word count of the words appearing in the files of  
my\_dataset: **comedies.txt, histories.txt, poems.txt, tragedies.txt**
- Writes the results to new files placed in the directory  
**dbfs:/FileStore/1\_Spark\_Core/my\_result**

# Databricks: An Online Platform for Data Engineering

- Without explaining the program itself, our code example

[Workspace/1\\_Spark\\_Core/6\\_Text\\_File\\_Examples/27\\_word\\_count.py](#)

```
27_word_count (Python)
MyCluster File View: Code Permissions Run All
194 # (i) Specifies the function F to be executed.
195 # (ii) Define any input parameter such this function F has to be called with
196 #
197 #
198 if __name__ == '__main__':
199     # 1. We use as many input arguments as needed
200     pass
201
202     # 2. Local or Databricks
203     local_False_databricks_True = True
204
205     # 3. We set the path to my_dataset and my_result
206     my_local_path =
207     my_databricks_path = "/"
208
209     my_dataset_dir = "FileStore/tables/1_Spark_Core/my_dataset/"
210     my_result_dir = "FileStore/tables/1_Spark_Core/my_result"
211
```

# Databricks: An Online Platform for Data Engineering

9. Running the program / Spark application leads to no visual results:

[Workspace/1\\_Spark\\_Core/6\\_Text\\_File\\_Examples/27\\_word\\_count.py](#)

27\_word\_count (Python)

MyCluster

File View: Code Permissions Run All Clear

```
1 | else:
2 |     dbutils.fs.rm(my_result_dir, True)
3 |
4 | # 5. We configure the Spark Context
5 | sc = pyspark.SparkContext.getOrCreate()
6 | sc.setLogLevel('WARN')
7 | print("\n\n\n")
8 |
9 |
10| # 6. We call to our main function
11| my_main(sc, my_dataset_dir, my_result_dir)
```

▶ (3) Spark Jobs

Command took 11.98 seconds -- by Ignacio.Castineiras@cit.ie at 9/27/2019, 6:00:52 PM on MyCluster

# Databricks: An Online Platform for Data Engineering

9. However, the folder **my\_result** in the DBFS has been populated:

The screenshot shows the Databricks Data browser interface. On the left, there's a sidebar with icons for Home, Workspace, Recents, Data (which is highlighted with a red box), and Clusters. The main area is titled "Create New Table (Python)" and shows the "Data source" tab selected (DBFS). The "DBFS" tab is highlighted with a blue border. Below that, it says "Select a file from DBFS". The left pane shows a tree view with "jars" and "tables" under "tables". The right pane shows a list of files and folders: "1\_Spark\_Core", "2\_Spark\_Streaming", "3\_Spark\_SQL", "4\_Spark\_GraphsLib", "5\_Spark\_MLlib", "my\_result\_merge.py", "my\_dataset", "my\_result" (highlighted with a red box), and a folder containing "\_SUCCESS" and four "part-00000" files (highlighted with a red box).

File/Folder	Description
1_Spark_Core	Spark Core library
2_Spark_Streaming	Spark Streaming library
3_Spark_SQL	Spark SQL library
4_Spark_GraphsLib	Spark GraphX library
5_Spark_MLlib	Spark MLlib library
my_result_merge.py	Script for merging results
my_dataset	Dataset file
my_result	Result folder (highlighted)
_SUCCESS	Success marker file
part-00000	Partition 00000
part-00001	Partition 00001
part-00002	Partition 00002
part-00003	Partition 00003

# Databricks: An Online Platform for Data Engineering

9. Unfortunately we cannot visualise the content of these DBFS files.

The screenshot shows the Databricks Data interface. On the left, there's a sidebar with icons for Home, Workspace, Recents, Data (which is highlighted with a red box), and Clusters. The main area is titled "Create New Table (Python)". Under "Data source", the "DBFS" tab is selected (also highlighted with a red box). The "Select a file from DBFS" section shows a tree view of files. A red box highlights three specific items: "1\_Spark\_Core" under "tables", "my\_result" under "tables", and a folder containing "\_SUCCESS" and four "part-XXXX" files under "tables".

Create New Table (Python)

Data source

Upload File S3 DBFS Other Data Sources

Select a file from DBFS

jars

tables

1\_Spark\_Core

2\_Spark\_Streaming

3\_Spark\_SQL

4\_Spark\_GraphsLib

5\_Spark\_MLlib

my\_result\_merge.py

my\_dataset

my\_result

\_SUCCESS

part-00000

part-00001

part-00002

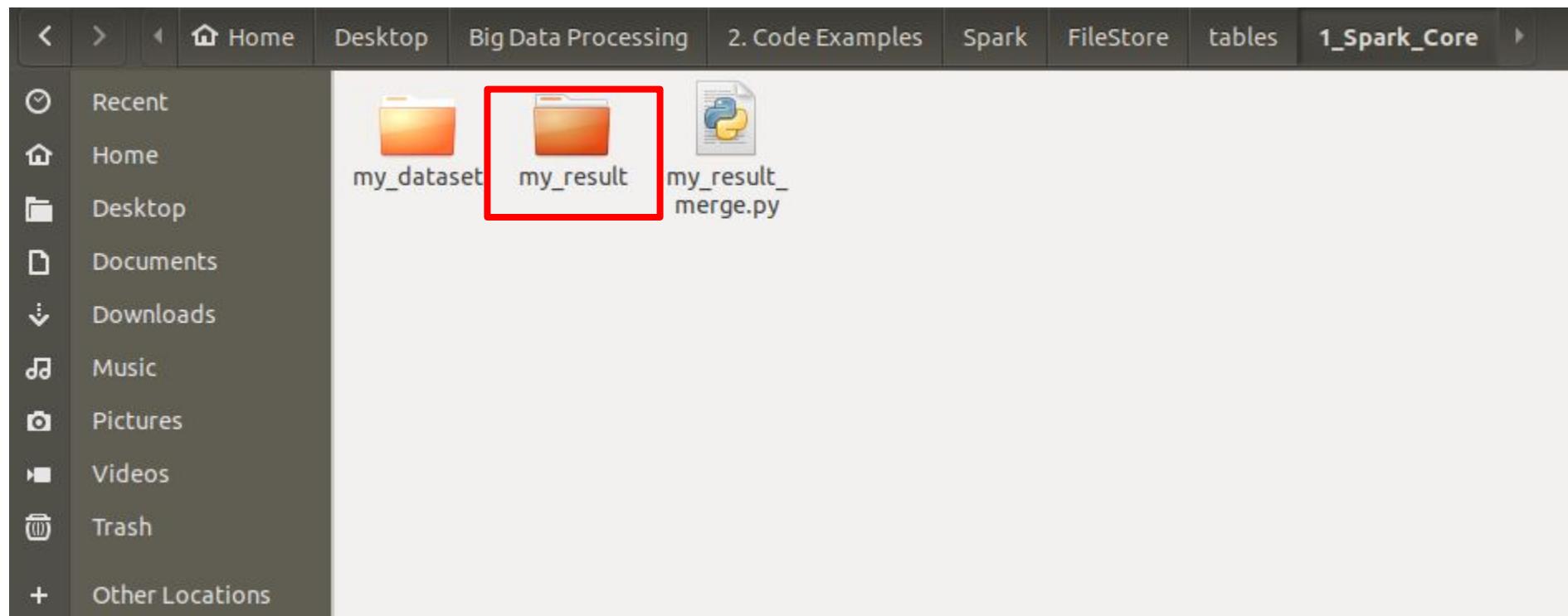
part-00003

# Databricks: An Online Platform for Data Engineering

9. But we can use the DBFS command cp to copy them to our local machine!

# Databricks: An Online Platform for Data Engineering

- As we can see, before running the command `cp` the local folder **my\_result** is empty.



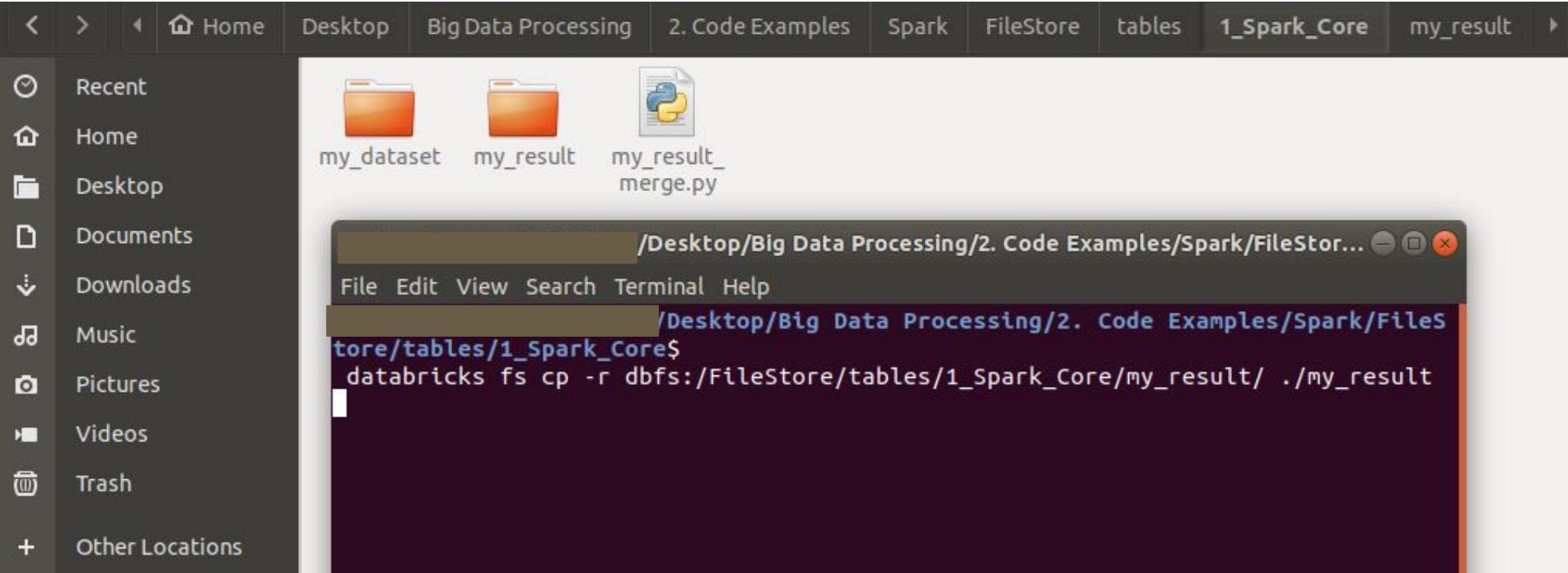
# Databricks: An Online Platform for Data Engineering

- As we can see, before running the command `cp` the local folder **my\_result** is empty.

The screenshot shows a file explorer window with a sidebar on the left containing icons for Recent, Home, Desktop, Documents, Downloads, Music, Pictures, Videos, Trash, and Other Locations. The main pane displays a folder icon with the text "Folder is Empty". The top navigation bar includes icons for back, forward, and search, followed by tabs for Home, Desktop, Big Data Processing, 2. Code Examples, Spark, FileStore, tables, 1\_Spark\_Core, and my\_result.

# Databricks: An Online Platform for Data Engineering

9. We run the command **cp** to copy the files.



# Databricks: An Online Platform for Data Engineering

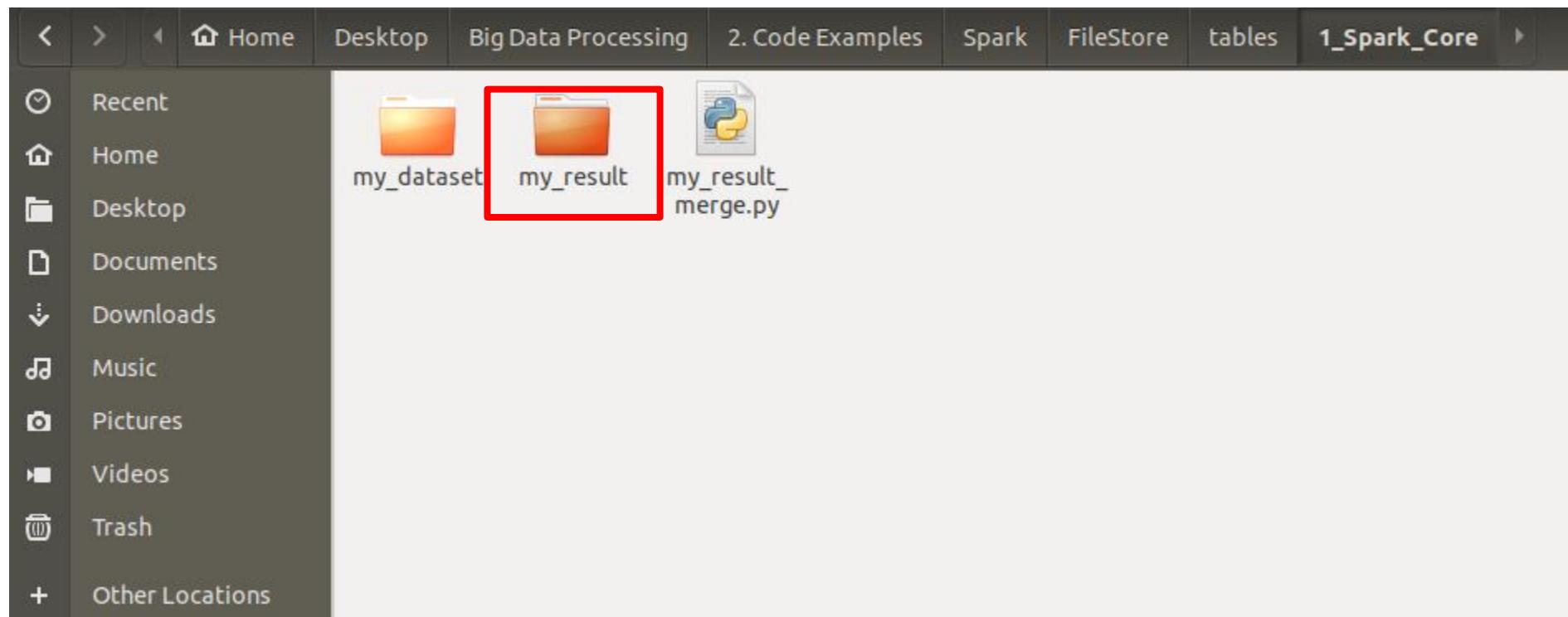
9. We run the command **cp** to copy the files.

The screenshot shows a desktop environment with a file manager window and a terminal window. The file manager window is open at `/Desktop/Big Data Processing/2. Code Examples/Spark/FileStore/tables/1_Spark_Core$`. It contains three items: `my_dataset`, `my_result`, and `my_result_merge.py`. The terminal window is also at the same directory and shows the command:

```
~/Desktop/Big Data Processing/2. Code Examples/Spark/FileStore/tables/1_Spark_Core$ databricks fs cp -r dbfs:/FileStore/tables/1_Spark_Core/my_result/ ./my_result  
dbfs:/FileStore/tables/1_Spark_Core/my_result/_SUCCESS -> ./my_result/_SUCCESS  
dbfs:/FileStore/tables/1_Spark_Core/my_result/part-00000 -> ./my_result/part-000  
00  
dbfs:/FileStore/tables/1_Spark_Core/my_result/part-00001 -> ./my_result/part-000  
01  
dbfs:/FileStore/tables/1_Spark_Core/my_result/part-00002 -> ./my_result/part-000  
02  
dbfs:/FileStore/tables/1_Spark_Core/my_result/part-00003 -> ./my_result/part-000  
03
```

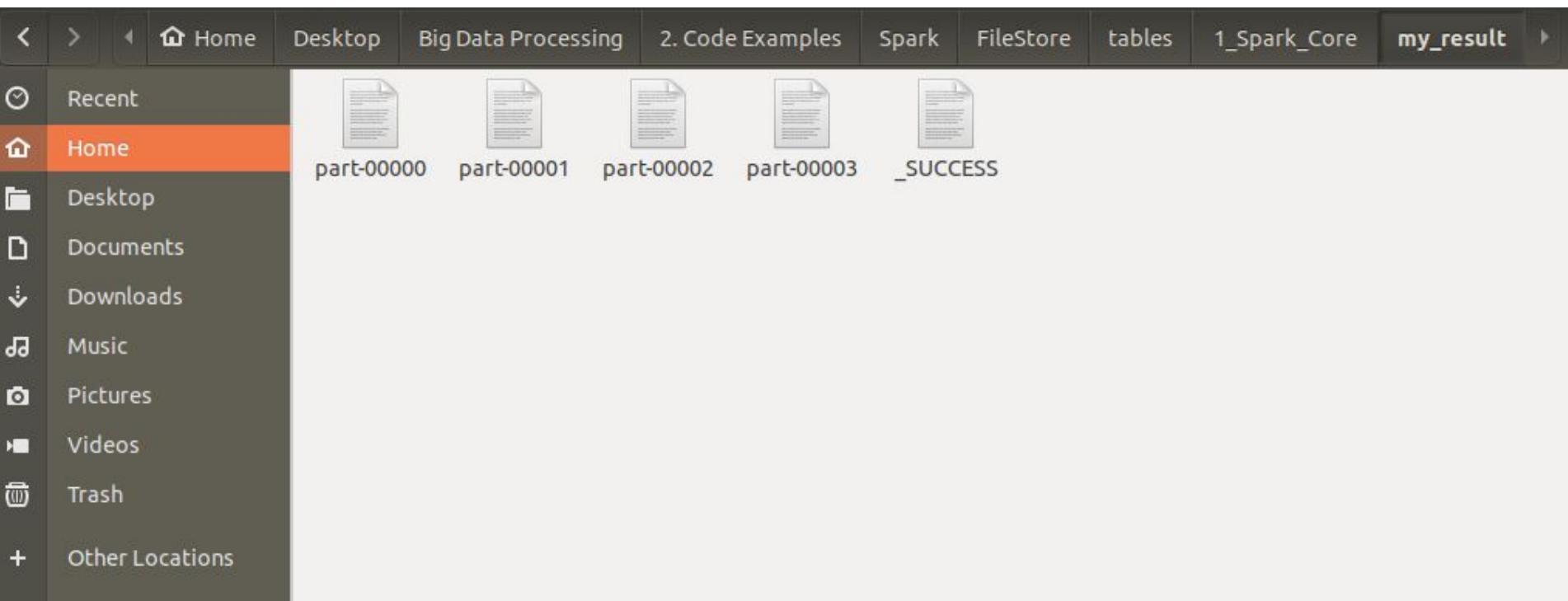
# Databricks: An Online Platform for Data Engineering

9. We run the command **cp** to copy the files.



# Databricks: An Online Platform for Data Engineering

9. And now that the solution files of running the Spark Application are back to our computer we can open them to see their content.



# Databricks: An Online Platform for Data Engineering

9. And now that the solution files of running the Spark Application are back to our computer we can open them to see their content.



The screenshot shows a Databricks notebook interface. The top navigation bar has 'Open' and '+ New' buttons. The header displays the file path: `~/Desktop/Big Data Processing/2. Code Examples/Spark/FileStore/tables/1_Spark_Core/my_result` and the part number `part-00000`. The main content area contains the following list of tuples:

```
('the', 29831)
('and', 27499)
('i', 21411)
('to', 20375)
('of', 18503)
('a', 15342)
('you', 13995)
('my', 12952)
('in', 11689)
('that', 11496)
('is', 9545)
('not', 8849)
('with', 8253)
```

# Databricks: An Online Platform for Data Engineers

*Databricks provides much more functionality, and I encourage you to explore it.*

*That being said, the one covered in these slides represents all the functionality we need to use for this module.*

# Outline

1. Introduction to Apache Spark.
2. A Spark Application: General Overview.
3. Two Different Roles: IT Manager vs. Data Engineer.
4. Databricks: An Online Platform for Data Engineers.

Thank you for your attention!