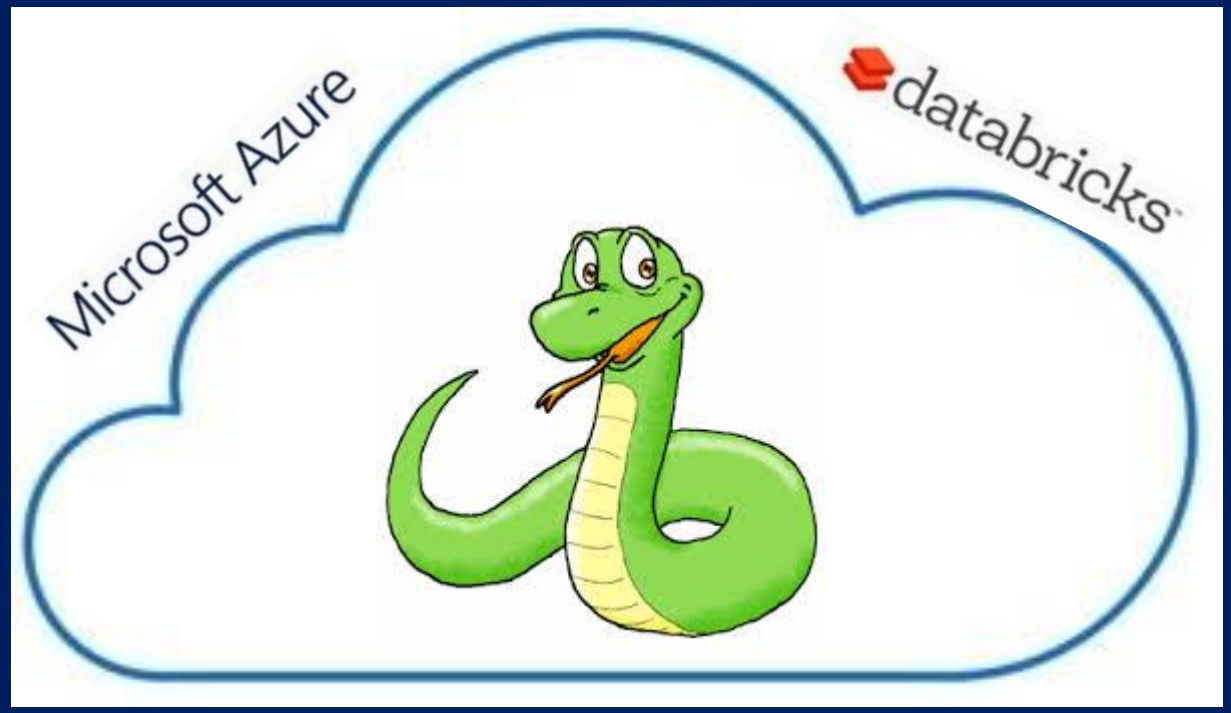
Microsoft

Azure Databricks with Python: Deep Dive



Bryan Cafferky
Data Solutions Enabler

APACHE SPARK

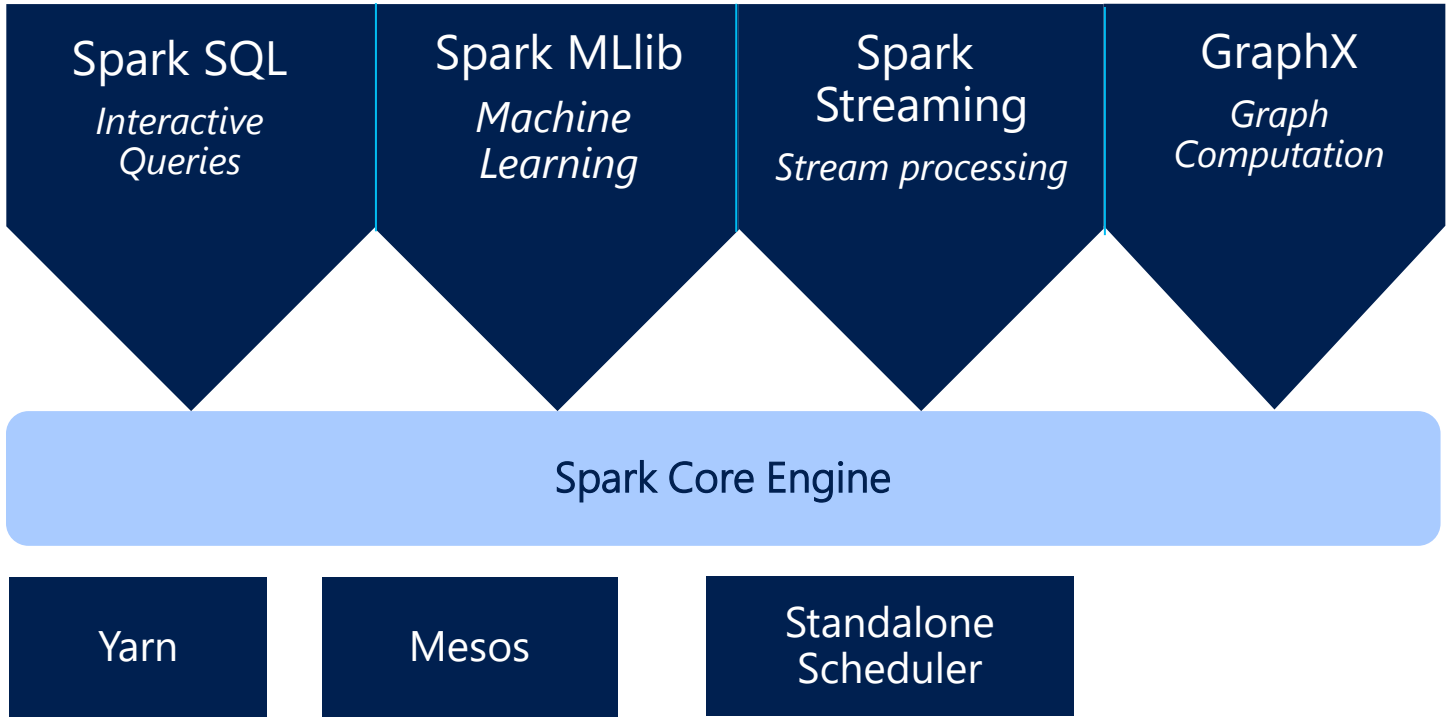
An unified, open source, parallel, data processing framework for Big Data Analytics

Python is not Supported by Spark (directly)

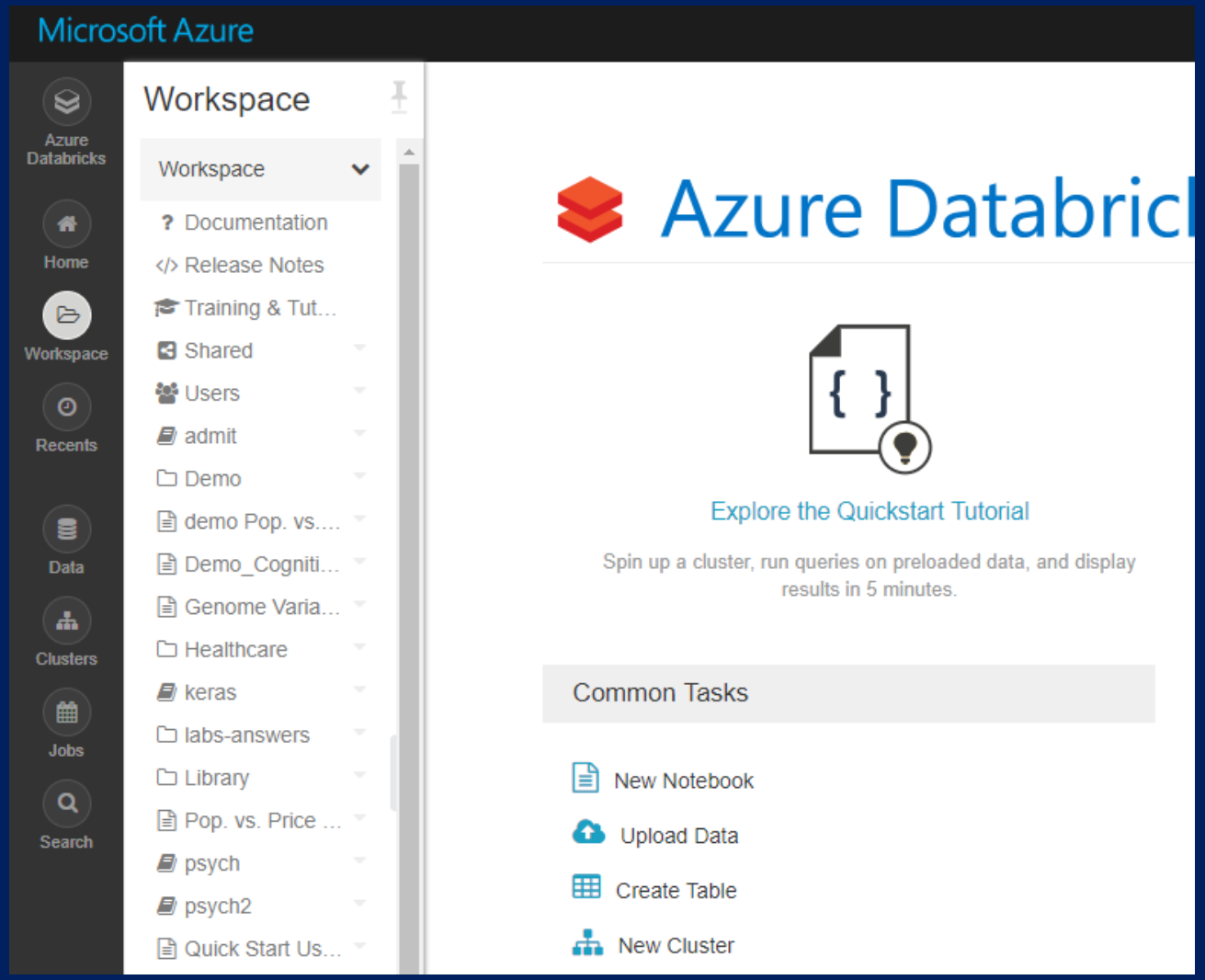
Spark Unifies:

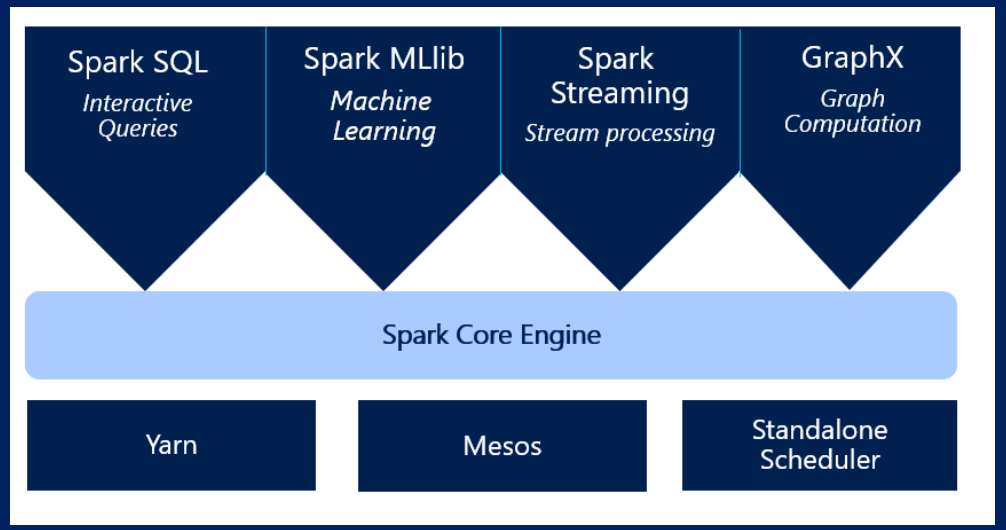
- Batch Processing
- Interactive SQL
- Real-time processing
- Machine Learning
- Deep Learning
- Graph Processing

The PySpark Package Wraps the Spark API

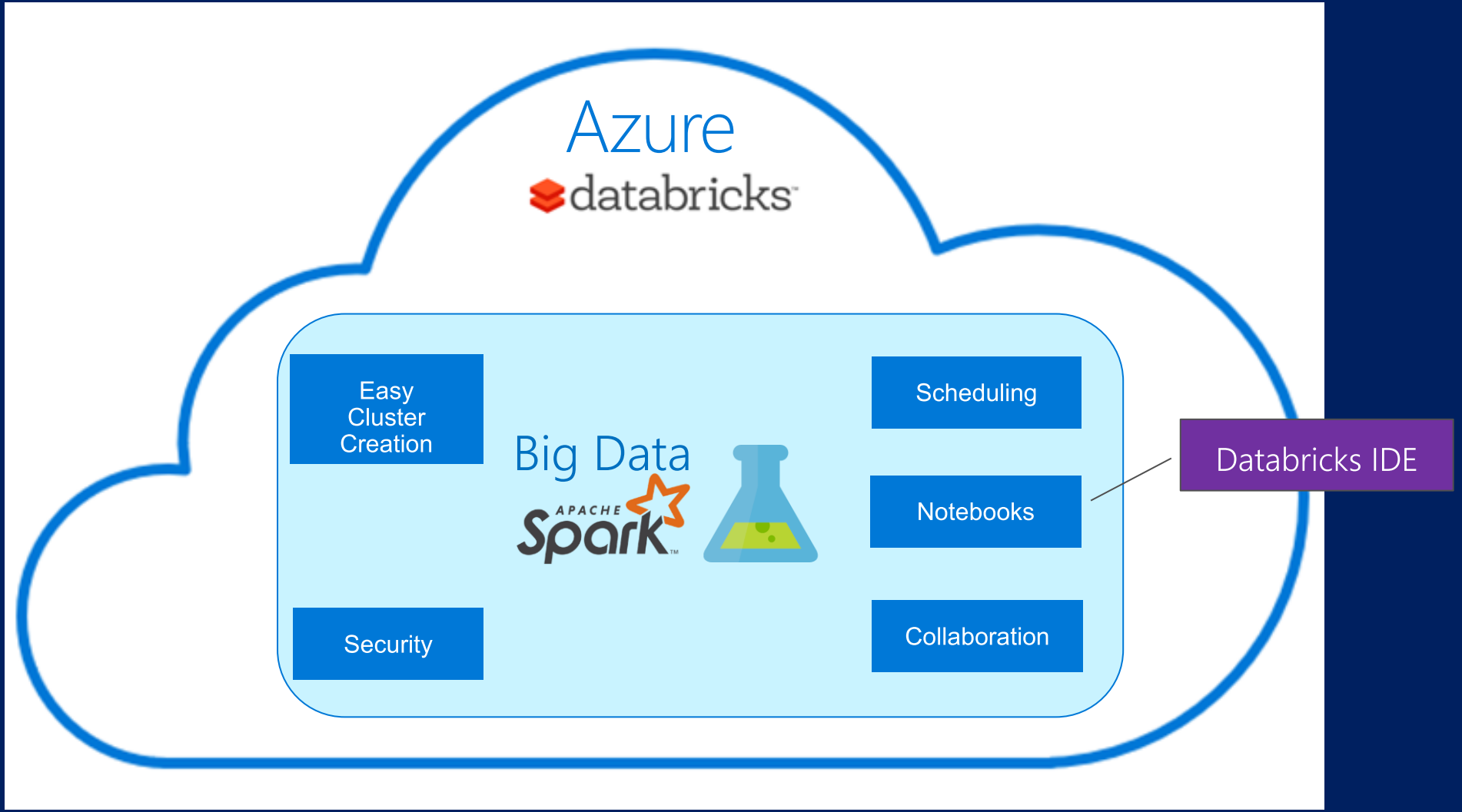


Azure Databricks Python Language Deep Dive





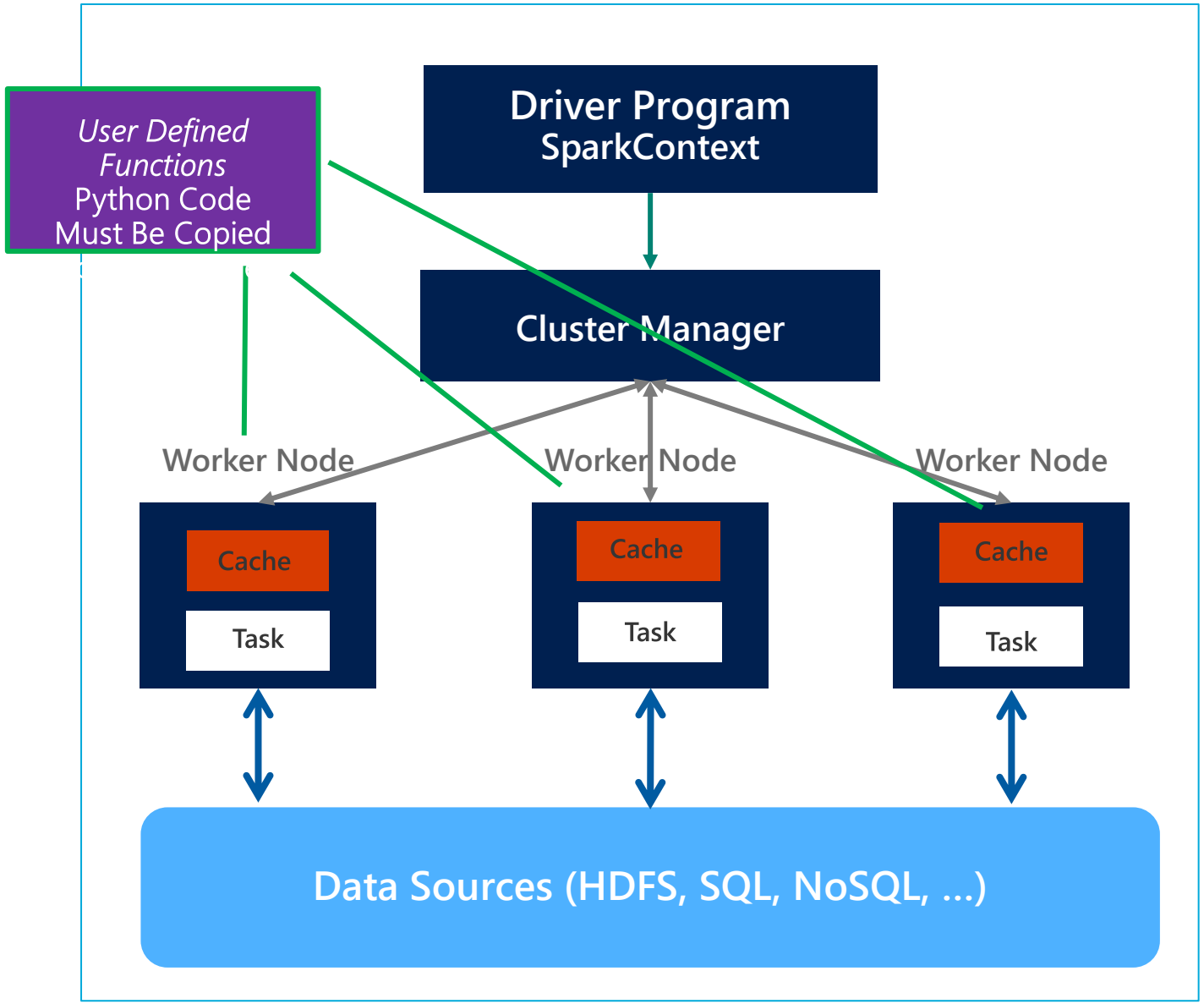
Microsoft


Azure
databricks
Easy Cluster Creation
Big Data
APACHE Spark
Scheduling
Notebooks
Collaboration
Security
Databricks IDE

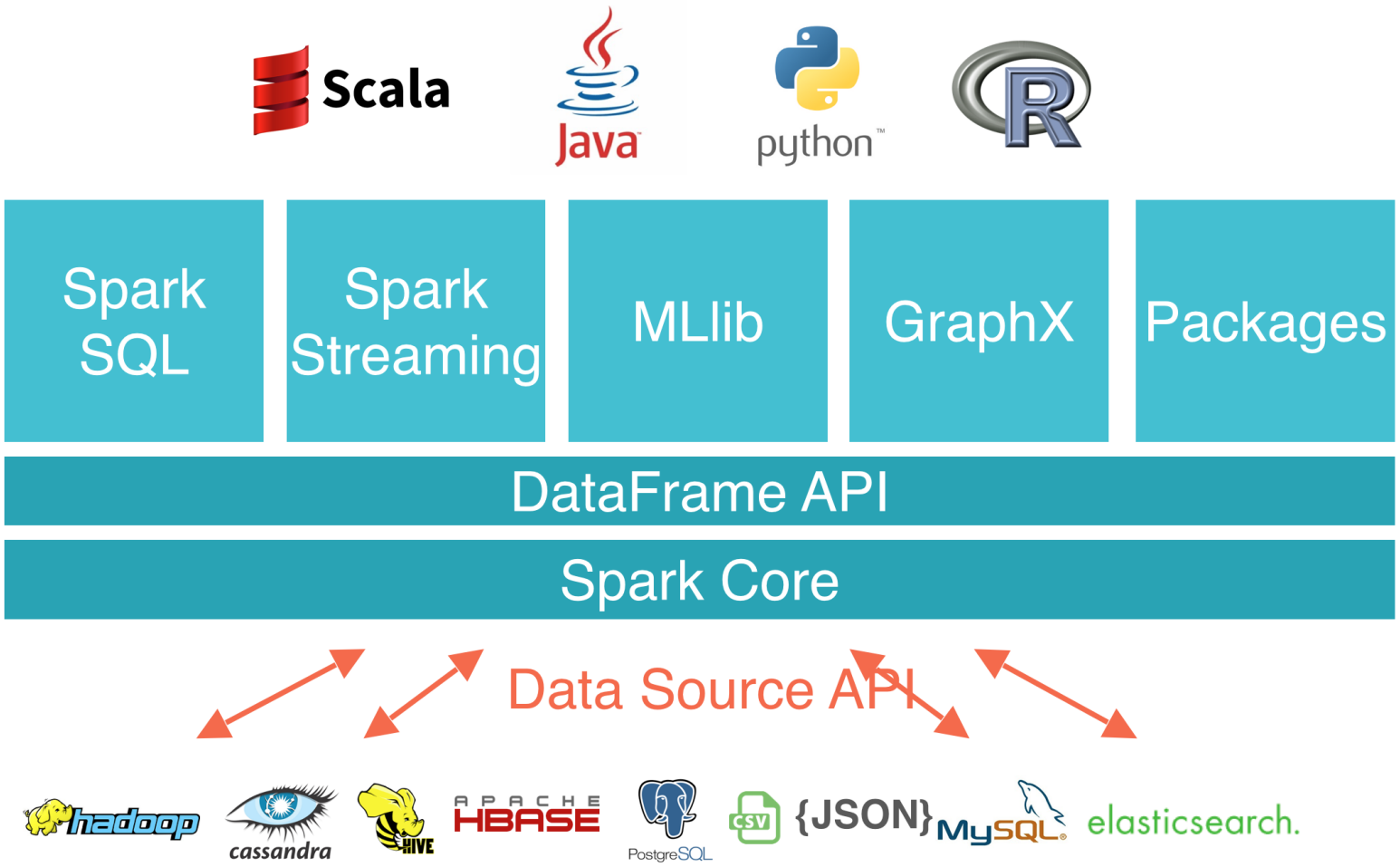
GENERAL SPARK CLUSTER ARCHITECTURE

- 'Driver' runs the user's 'main' function and executes the various parallel operations on the worker nodes.
- The results of the operations are collected by the driver
- The worker nodes read and write data from/to Data Sources including HDFS.
- Worker node also cache transformed data in memory as RDDs (Resilient Data Sets).
- Worker nodes and the Driver Node execute as VMs in public clouds (AWS, Google and Azure).

Nodes
Run JVM



Azure Databricks Python Language Deep Dive



Spark RDD to Dataframe – Win/Win

Originally had to use RDD

Dataframe Support Added in 1.x

Native Language Paradigm/Feel

Easier to Read

Performs much better!

Translates to SQL which can be optimized and tuned

We will focus on the Dataframe API

What is an API?

Application Programming Interface

Exposes a Service so it Can Be Called

Provides a Standard Way to Call the Exposed Service

Implementation Details are Hidden from the Calling Program

The Calling Program Can Hide the API Details from the User

What Does a Spark API Do?

Load Data for Use By Spark

Read and Manipulate Data in Spark

Push Processing to the Spark Cluster Nodes

Do Work on the Head Node

Retain the Feel and Paradigm of the Calling Language

Obfuscating Spark with the Language API

Spark API

Scala

```scala
val spark = new SparkContext()

val lines    = spark.textFile("hdfs://docs/")    // RDD[String]
val nonEmpty = lines.filter(l => l.nonEmpty())   // RDD[String]

val count = nonEmpty.count
```

Java 8

```java
SparkContext spark = new SparkContext();

JavaRDD<String> lines    = spark.textFile("hdfs://docs/")
JavaRDD<String> nonEmpty = lines.filter(l -> l.length() > 0);

long count = nonEmpty.count();
```

Python

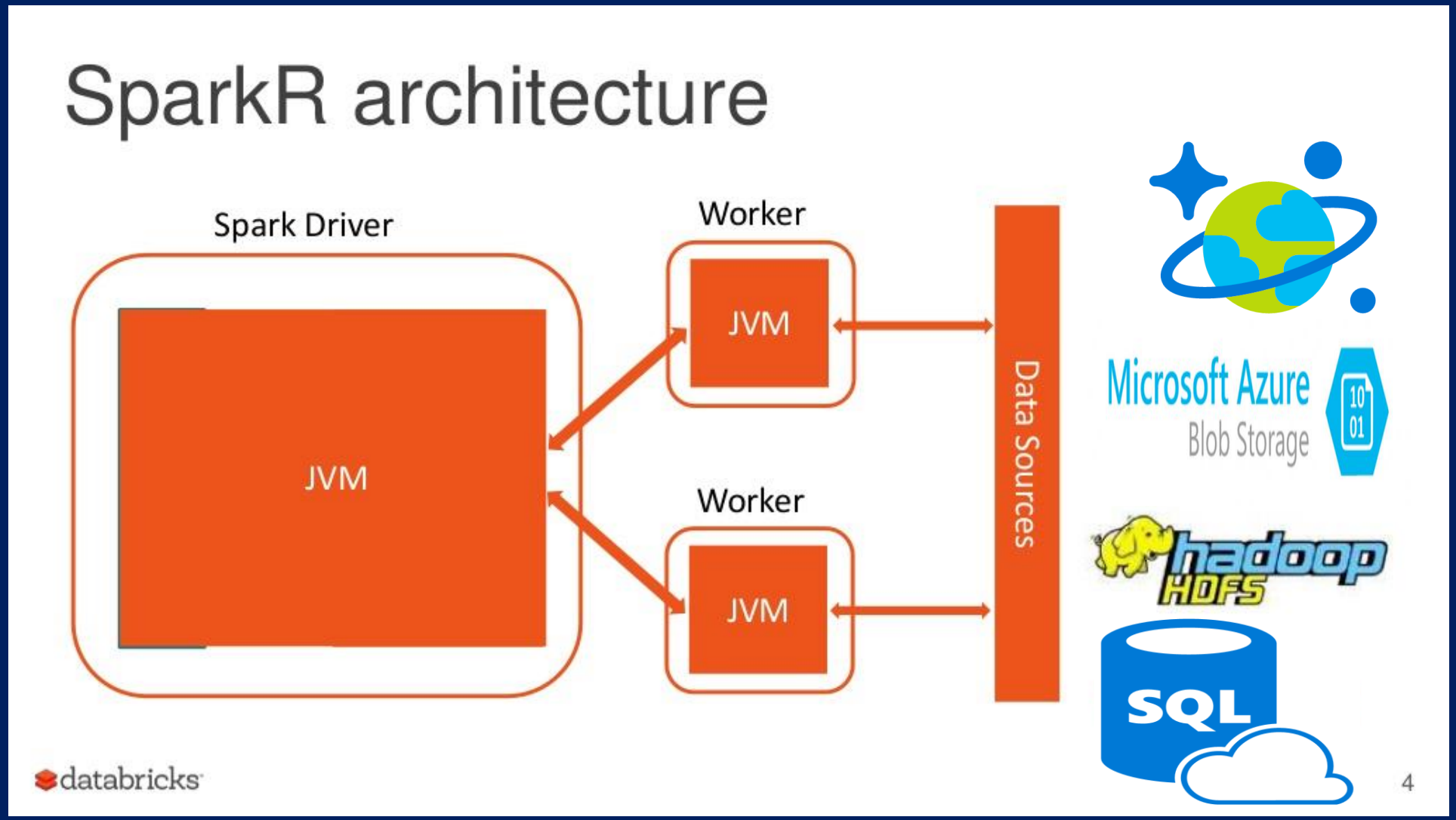
```python
spark = SparkContext()

lines = spark.textFile("hdfs://docs/")
nonEmpty = lines.filter(lambda line: len(line) > 0)

count = nonEmpty.count()
```

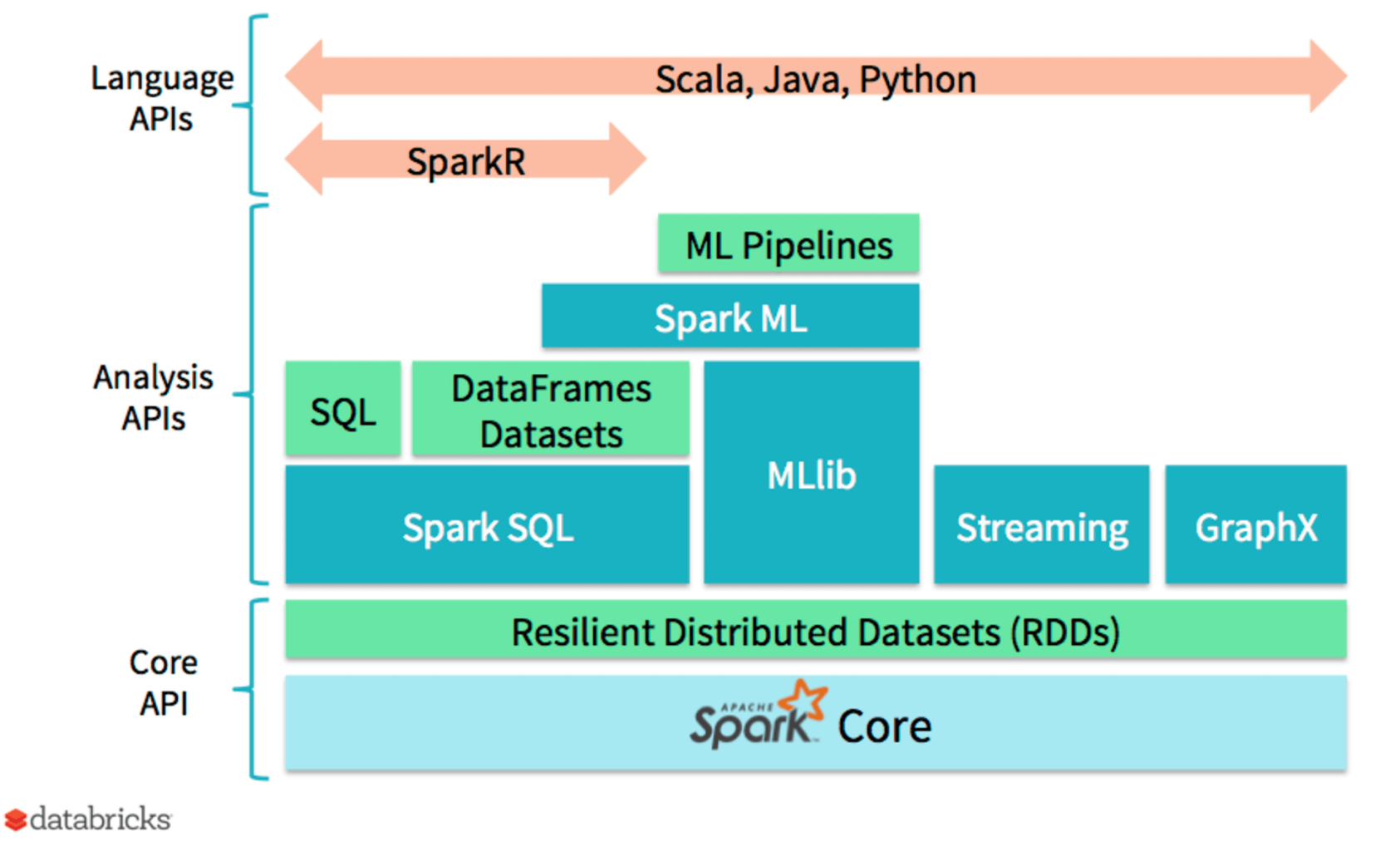
https://www.slideshare.net/frodriguezolivera/apache-spark-streaming

Apache Spark API



https://www.slideshare.net/databricks/parallelizing-existing-r-packages-with-sparkr

SparkR – What Do We Get?



https://databricks.com/blog/2016/12/28/10-things-i-wish-i-knew-before-using-apache-sparkr.html

Scaling Machine Learning

Open Source Machine Learning Libraries Do Not Scale

- Input Data Size Limits
- Do Not Support Parallel Processing
- Are Not Multithreaded

You Will Need to Change Your Model Library/Function

Azure Databricks Python Language Deep Dive

Apache Spark API

PySpark 2.3.2 documentation »

Table of Contents

pyspark package
- Subpackages
- Contents
  - SparkConf
  - SparkContext
  - SparkFiles
  - RDD
  - StorageLevel
  - Broadcast
  - Accumulator
  - AccumulatorParam
  - MarshalSerializer
  - PickleSerializer
  - StatusTracker
  - SparkJobInfo
  - SparkStageInfo
  - Profiler
  - BasicProfiler
  - TaskContext E

Previous topic

Welcome to Spark Python API Docs!

Next topic

pyspark.sql module

This Page

Show Source

Quick search

pyspark package

Subpackages

- pyspark.sql module
- pyspark.streaming module
- pyspark.ml package
- pyspark.mllib package

Contents

PySpark is the Python API for Spark.

Public classes:

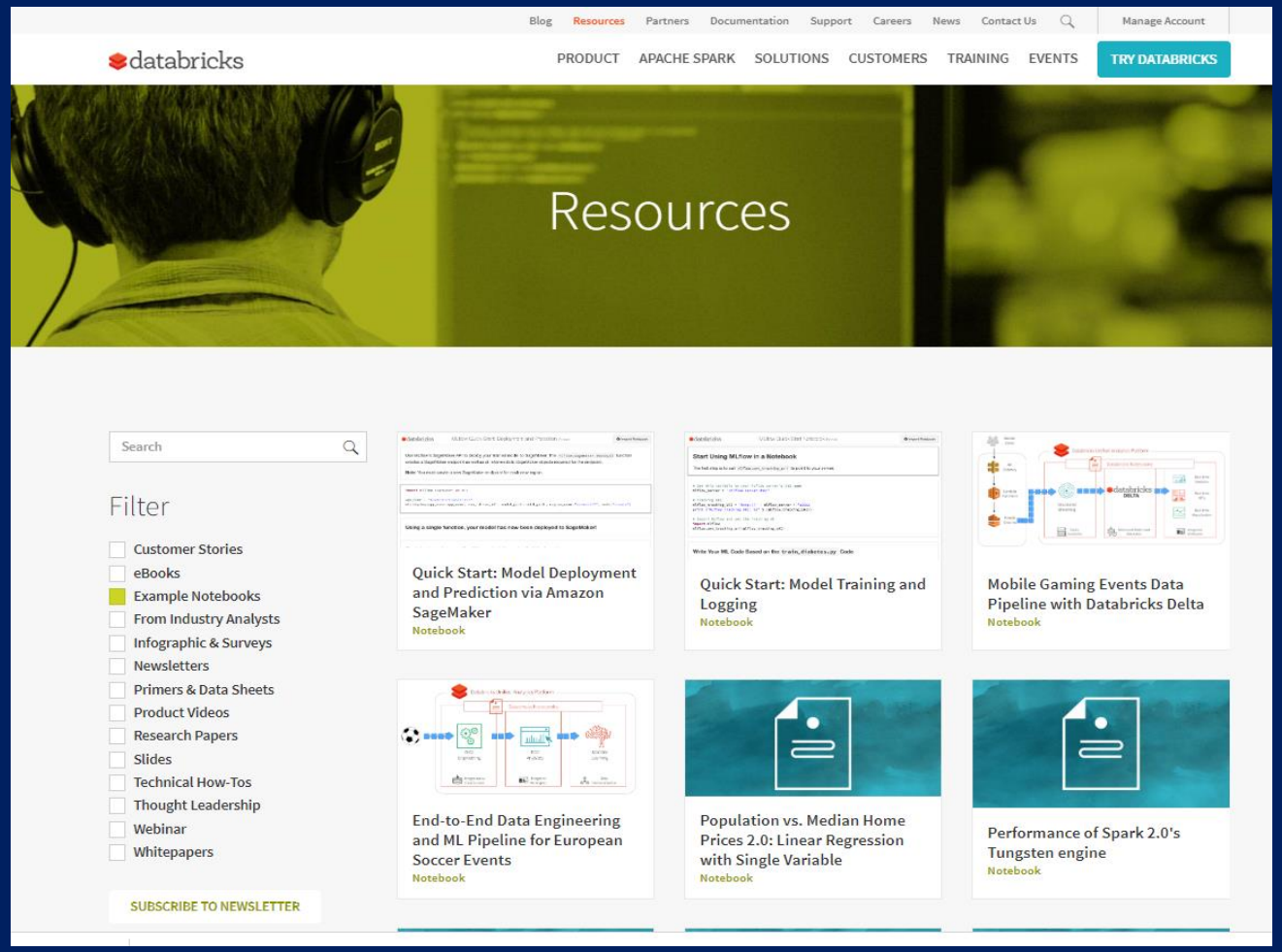
- **SparkContext**:
  Main entry point for Spark functionality.
- **RDD**:
  A Resilient Distributed Dataset (RDD), the basic abstraction in Spark.
- **Broadcast**:
  A broadcast variable that gets reused across tasks.
- **Accumulator**:
  An "add-only" shared variable that tasks can only add values to.
- **SparkConf**:
  For configuring Spark.
- **SparkFiles**:
  Access files shipped with jobs.
- **StorageLevel**:
  Finer-grained cache persistence levels.
- **TaskContext**:
  Information about the current running task, available on the workers and experimental.

Supported APIs

- ML is for Pipelines
- MLLib is the Spark ML Library

https://spark.apache.org/docs/latest/api/python/pyspark.html#subpackages

Getting Started on Azure Databricks



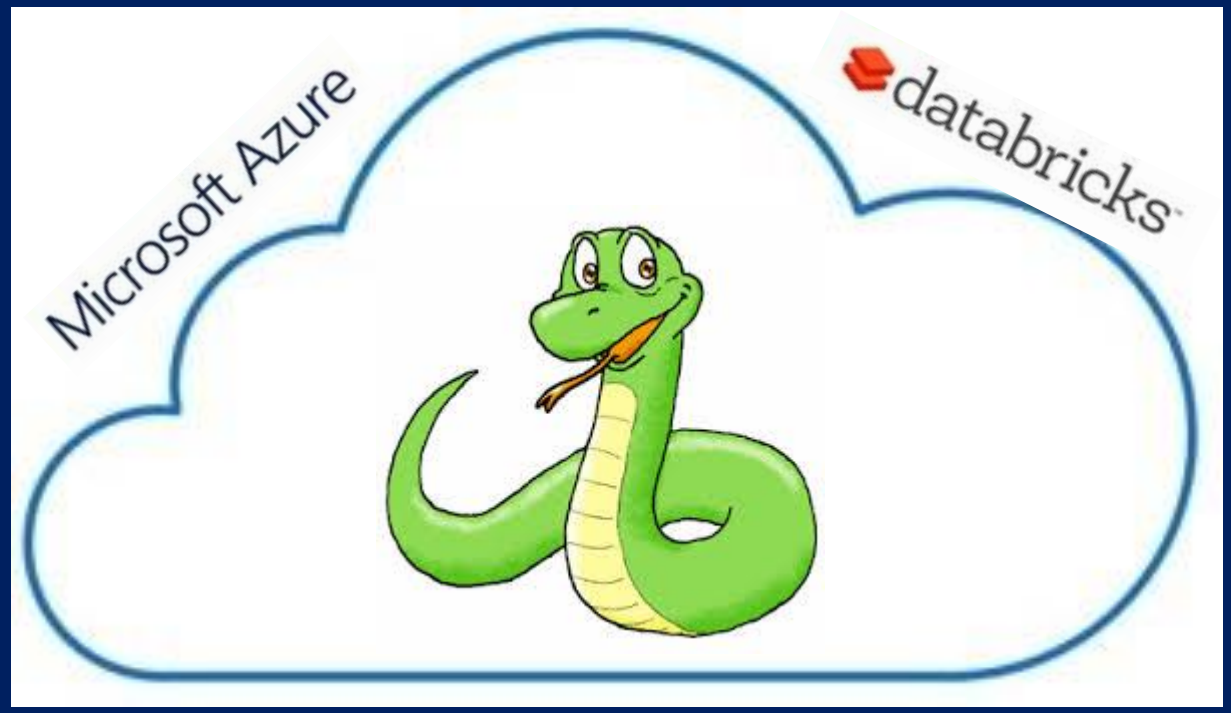
https://docs.azuredatabricks.net/index.html

https://databricks.com/resources/type/example-notebooks

Azure Databricks R Deep Dive

Microsoft

Azure Databricks with Python: Deep Dive



Bryan Cafferky
Data Solutions Enabler