Chris Snow
Big Data Architect – Cloud Data Services
@csnow_uk

Analysing Big Data with R using Apache Spark and DSX

Disclaimer

THE INFORMATION CONTAINED IN THIS PRESENTATION IS PROVIDED FOR INFORMATIONAL PURPOSES ONLY. WHILE EFFORTS WERE MADE TO VERIFY THE COMPLETENESS AND ACCURACY OF THE INFORMATION CONTAINED IN THIS PRESENTATION, IT IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. IN ADDITION, THIS INFORMATION IS BASED ON IBM'S CURRENT PRODUCT PLANS AND STRATEGY, WHICH ARE SUBJECT TO CHANGE BY IBM WITHOUT NOTICE. IBM SHALL NOT BE RESPONSIBLE FOR ANY DAMAGES ARISING OUT OF THE USE OF, OR OTHERWISE RELATED TO, THIS PRESENTATION OR ANY OTHER DOCUMENTATION. NOTHING CONTAINED IN THIS PRESENTATION IS INTENDED TO, NOR SHALL HAVE THE EFFECT OF, CREATING ANY WARRANTIES OR REPRESENTATIONS FROM IBM (OR ITS SUPPLIERS OR LICENSORS), OR ALTERING THE TERMS AND CONDITIONS OF ANY AGREEMENT OR LICENSE GOVERNING THE USE OF IBM PRODUCTS AND/OR SOFTWARE.

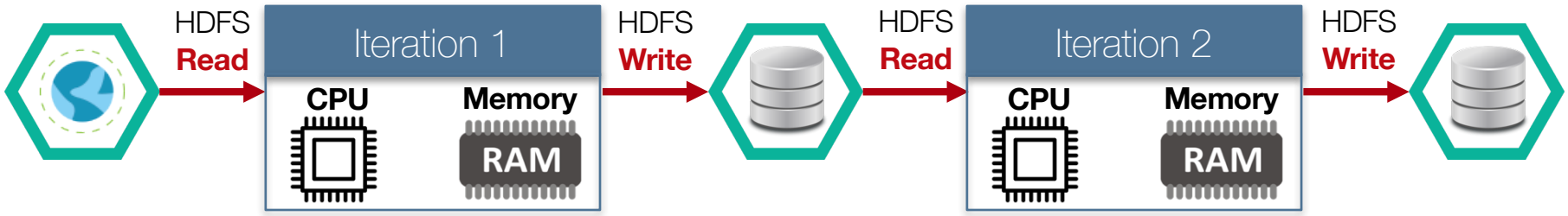
IBM's statements regarding its plans, directions, and intent are subject to change or withdrawal without notice at IBM's sole discretion. Information regarding potential future products is intended to outline our general product direction and it should not be relied on in making a purchasing decision. The information mentioned regarding potential future products is not a commitment, promise, or legal obligation to deliver any material, code or functionality. Information about potential future products may not be incorporated into any contract. The development, release, and timing of any future features or functionality described for our products remains at our sole discretion.

IBM Analytics for Apache Spark

What is Spark?

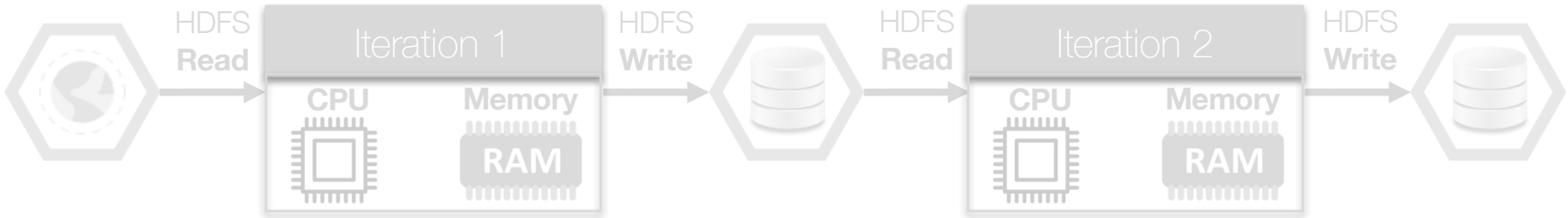
Motivation for Apache Spark

- **Traditional Approach: MapReduce jobs for complex jobs, interactive query, and online event-hub processing involves lots of (slow) disk I/O**

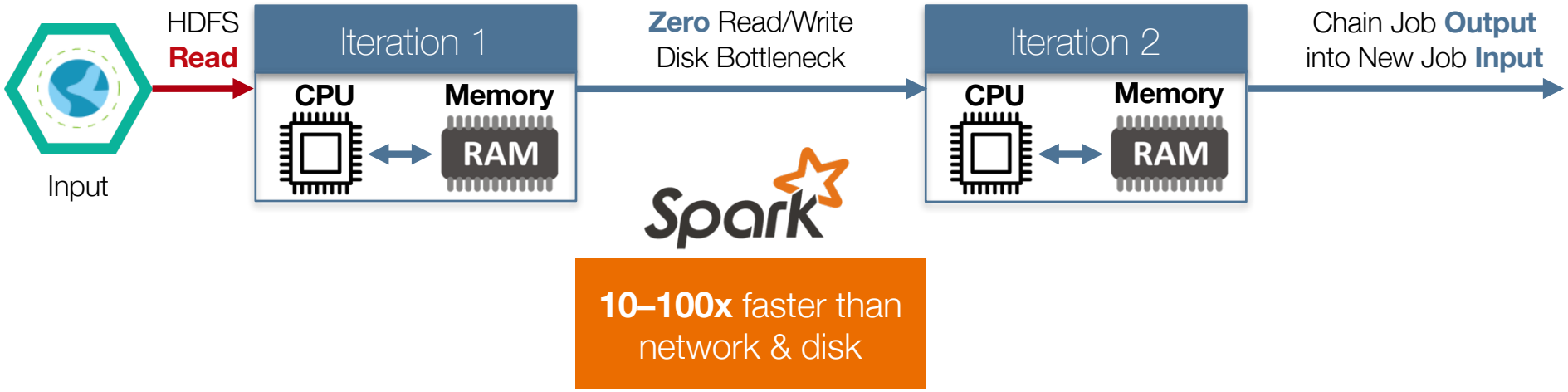


Motivation for Apache Spark

- **Traditional Approach: MapReduce jobs for complex jobs, interactive query, and online event-hub processing involves lots of (slow) disk I/O**



- **Solution: Keep data in-memory with a new distributed execution engine**



Positioning Hadoop and Spark

Spark	Spark	Hadoop MapReduce
Storage	No built-in storage (in-memory)	On-disk only
Operations	Map, Reduce, Join, Sample, etc.	Map and Reduce
Execution Model	Batch, interactive, streaming	Batch only
Programming Environments	Python, Scala, Java, and R	Java only

- **Spark is a unified platform for data integration**
  → Contrast with the many distinct distributions & tools for Hadoop
- **Spark follows lazy evaluation of execution graphs**
  → Optimizes jobs, reduces wait states, and allows easier pipelining of tasks
- **Spark lowers the resource overhead for starting or shuffling jobs**
  → Less expensive than MapReduce

IBM Analytics for Apache Spark

Only IBM brings strength in enterprise, scale, and a managed offering to the Spark market

Performant Architecture

Productive Workflows

Leverages Existing Investments

Continually Improving

Spark as a service

- **Fully-managed** & **secured** Spark environment, accessible on-demand or via reserved instances
- In-memory architecture greatly reduces disk I/O
  - **20-100x** faster for common tasks
- Analytic workflows across a multitude of sources
  - Simplified but powerful syntax (**~5x less code**)
  - Integrates with **SQL, Java**, **Python**, **Scala**, etc.
- No lock-in: 100% open source Spark
  - **Spark v1.6+** since February 2016
  - Continually updated apace Spark ecosystem
- Pay-as-you-go or Dedicated deployment options

IBM Analytics for Apache Spark

Open Source Community and IBM

History of Apache Spark

- **Started as a research project in 2009, open source in 2010**
  - General purpose cluster computing system
  - Batch oriented processing
  - **Immutable** Resilient Distributed Datasets (RDDs)
- **Apache incubator project in June 2013**
  - Apache top level project Feb 27, 2014
- **Current version 1.6.1 (March 9, 2016)**
  - Spark **2.0** in **Preview** release (May 26, 2016)
  - Requires Scala 2.10.x, Maven
  - Languages supported: Java, Scala, Python, R (Java 7+, Python 2.6+, R 3.1+)
  - May need additional libraries for Python (e.g. numpy)

- Spark 1.6.1 (Mar 09 2016)
- Spark 1.6.0 (Jan 04 2016)
- Spark 1.5.2 (Nov 09 2015)
- Spark 1.5.1 (Oct 02 2015)
- Spark 1.5.0 (Sep 09 2015)
- Spark 1.4.1 (Jul 15 2015)
- Spark 1.4.0 (Jun 11 2015)
- Spark 1.3.1 (Apr 17 2015)
- Spark 1.3.0 (Mar 13 2015)
- Spark 1.2.2 (Apr 17 2015)
- Spark 1.2.1 (Feb 09 2015)
- Spark 1.2.0 (Dec 18 2014)
- Spark 1.1.1 (Nov 26 2014)
- Spark 1.1.0 (Sep 11 2014)
- Spark 1.0.2 (Aug 05 2014)
- Spark 1.0.1 (Jul 11 2014)
- Spark 1.0.0 (May 30 2014)
- Spark 0.9.2 (Jul 23 2014)
- Spark 0.9.1 (Apr 09 2014)
- Spark 0.9.0 (Feb 02 2014)
- Spark 0.8.1 (Dec 19 2013)

Spark Infused Across IBM Analytics Portfolio

Free and Open Data	▪ Analytics Exchange
Data Storage	▪ **On-Premises:** IBM Open Platform with Apache Hadoop (IOP), BigInsights, Neteeza, Cloudant, DB2, dashDB local and Informix ▪ **On-Cloud:** Cloudant, dashDB, Object Storage, SQL DB, BigInsights
Data Feeds, Load & Refinement	▪ DataWorks ▪ IBM Streams ▪ IBM Insights for Twitter ▪ IBM Insights for Weather
Analytics and Solutions	▪ IBM Analytics for Apache spark ▪ SPSS Modeler and Analytics Server ▪ Watson Analytics ▪ Watson Health ▪ IBM Commerce ▪ Data Science Experience
Learning Tools	▪ Big Data University

IBM DATA SCIENCE EXPERIENCE

ALL YOUR TOOLS IN ONE PLACE

IBM Data Science Experience provides an environment that brings together everything that a data scientist needs. It includes the most popular Open Source tools and IBM unique value-add functionalities with community and social features, integrated as a first class citizen to make data scientists more successful.

datascience.ibm.com

IBM Data Science Experience

- **"All your tools in one place" to make every data scientist successful and connected to the business**

Learn

Built-in learning to get started or go the distance with advanced tutorials

Create

The best of open source and IBM value-add to create state of the art data products

Collaborate

Community and social features that provide meaningful collaboration

Core Attributes of the Data Scientist Experience

IBM Data Science Experience

Community

- Find tutorials and datasets
- Connect with data scientists
- Ask questions
- Read articles and papers
- Fork and share projects

Open Source

- Code in Scala/Python/R/SQL
- Jupyter and Zeppelin* Notebooks
- RStudio IDE and Shiny apps
- **Apache Spark**
- Your favorite libraries

IBM Added Value

- Data Shaping/Pipeline UI *
- Auto-data preparation and modeling*
- Advanced Visualizations*
- Model management and deployment*
- Documented Model APIs*
- Spark as a Service *

Powered by IBM **Next Generation Platform** in the Cloud

* DSX product roadmap items

IBM Analytics for Apache Spark

Spark Core Engine

Spark Programming Languages

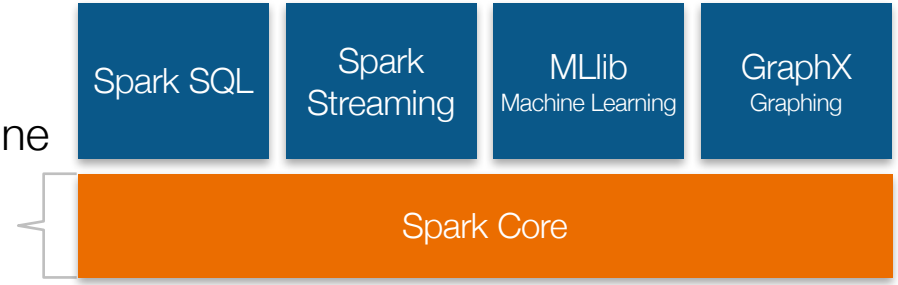
- **Scala**
  - Functional programming
  - Spark written in Scala
  - Scala compiles into Java byte code
- **Java**
  - New features in Java 8 makes for more compact coding (lambda expressions)
- **Python**
  - Always a bit behind Scala in functionality
- **R Statistical Language**
  - The latest addition to Spark

Language	2014	2015
Scala	84%	71%
Java	38%	31%
Python	38%	58%
R	N/A	18%

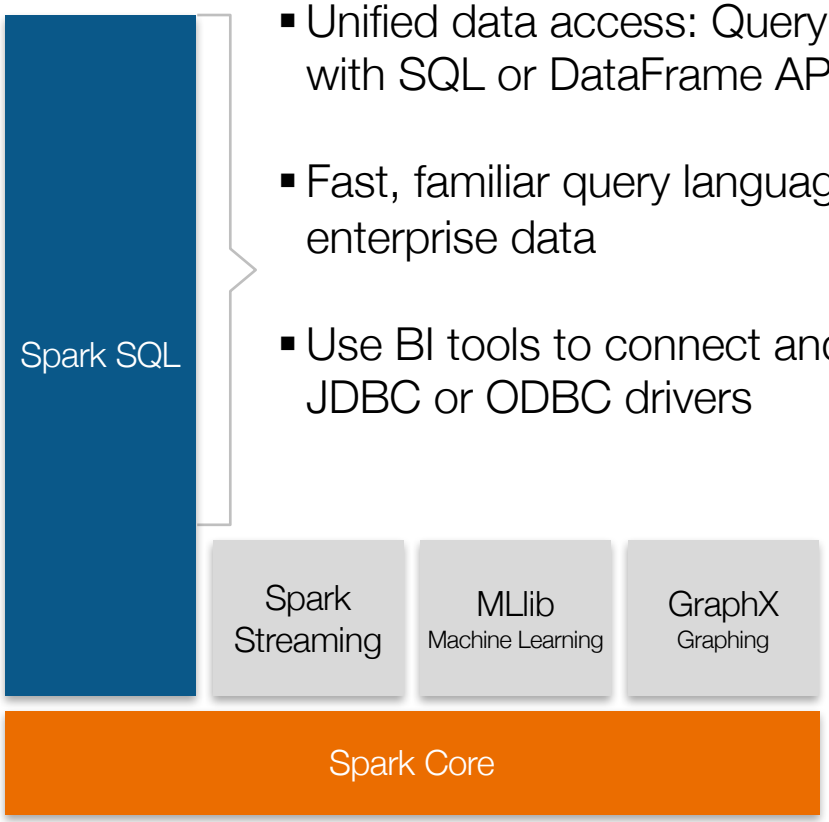
 Source: 2015 Databricks Spark Survey

Apache Spark Core functionality

- General compute engine
- Basic I/O functions
- Task dispatching
- Scheduling



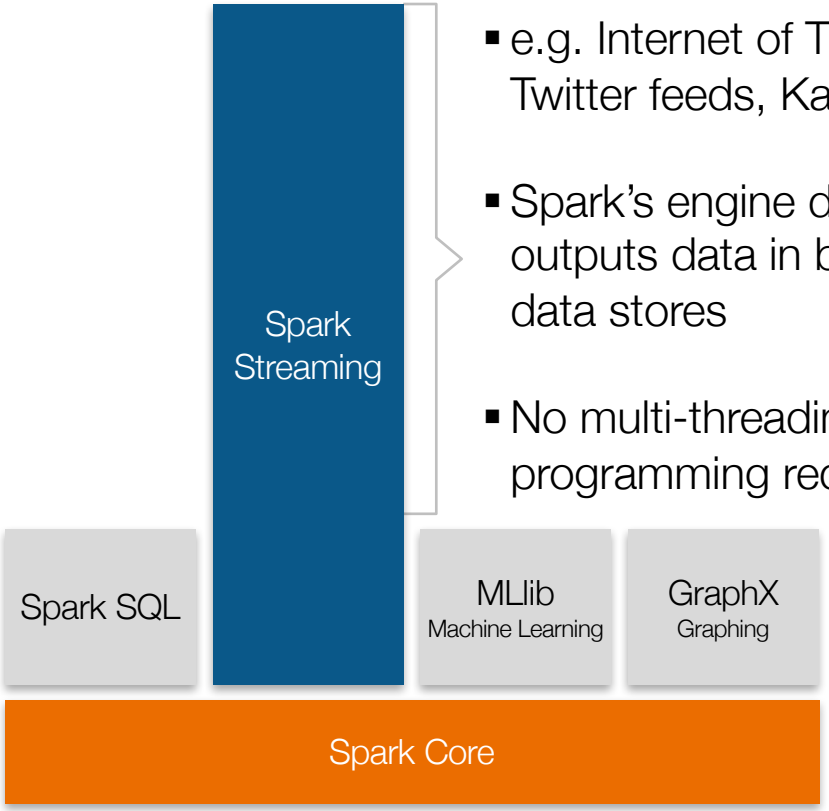
Select Libraries to Meet Use-Case Challenges



- Unified data access: Query structured data sets with SQL or DataFrame APIs
- Fast, familiar query language across all of your enterprise data
- Use BI tools to connect and query via JDBC or ODBC drivers

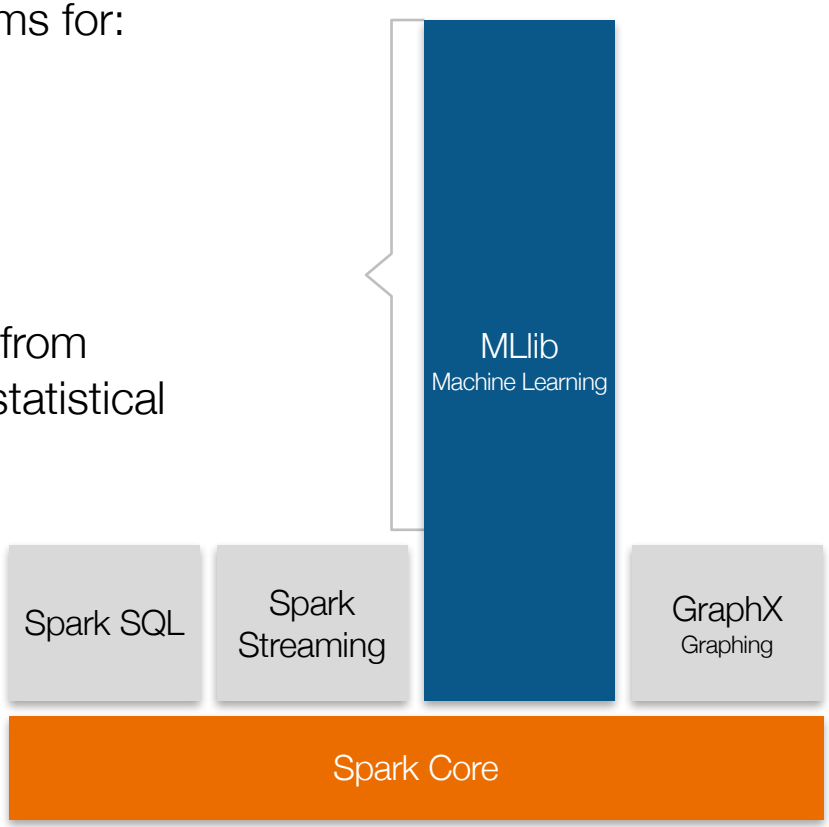
Select Libraries to Meet Use-Case Challenges

- Micro-batch event processing for near-real time analytics
- e.g. Internet of Things (IoT) devices, Twitter feeds, Kafka (event hub), etc.
- Spark's engine drives some action or outputs data in batches to various data stores
- No multi-threading or parallel process programming required



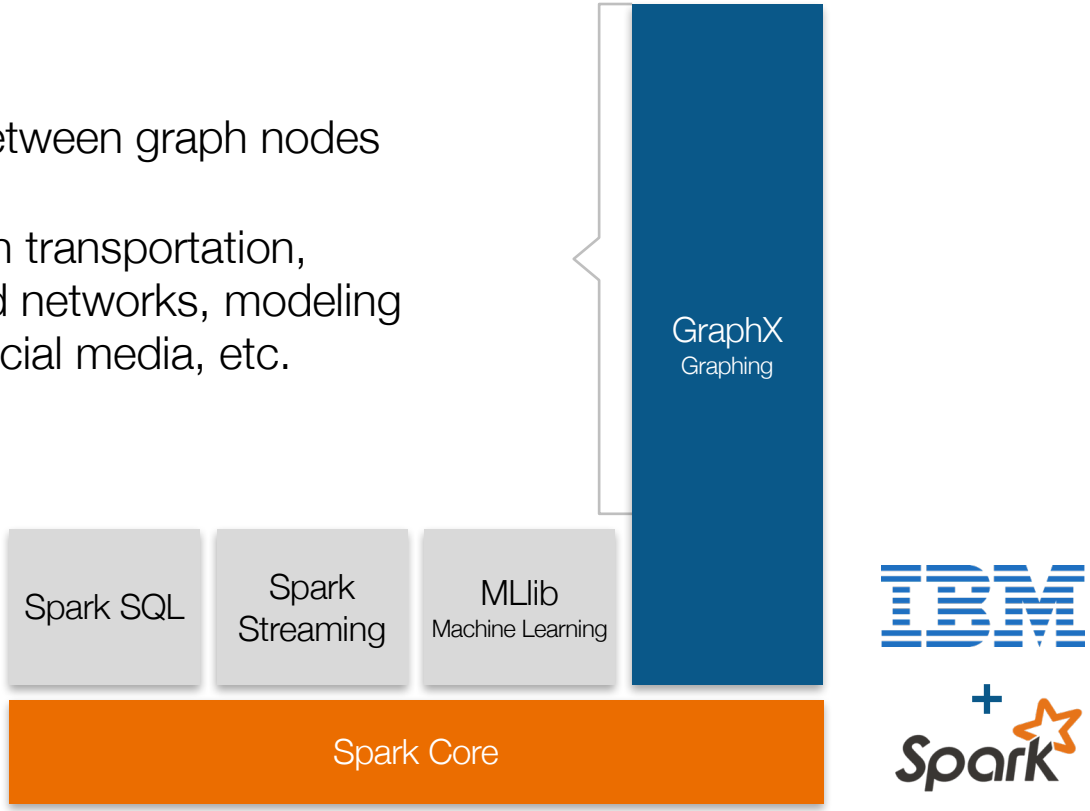
Select Libraries to Meet Use-Case Challenges

- Predictive and prescriptive analytics
- Machine learning algorithms for:
  - Clustering
  - Classification
  - Regression
  - etc.
- Smart application design from pre-built, out-of-the-box statistical and algorithmic models

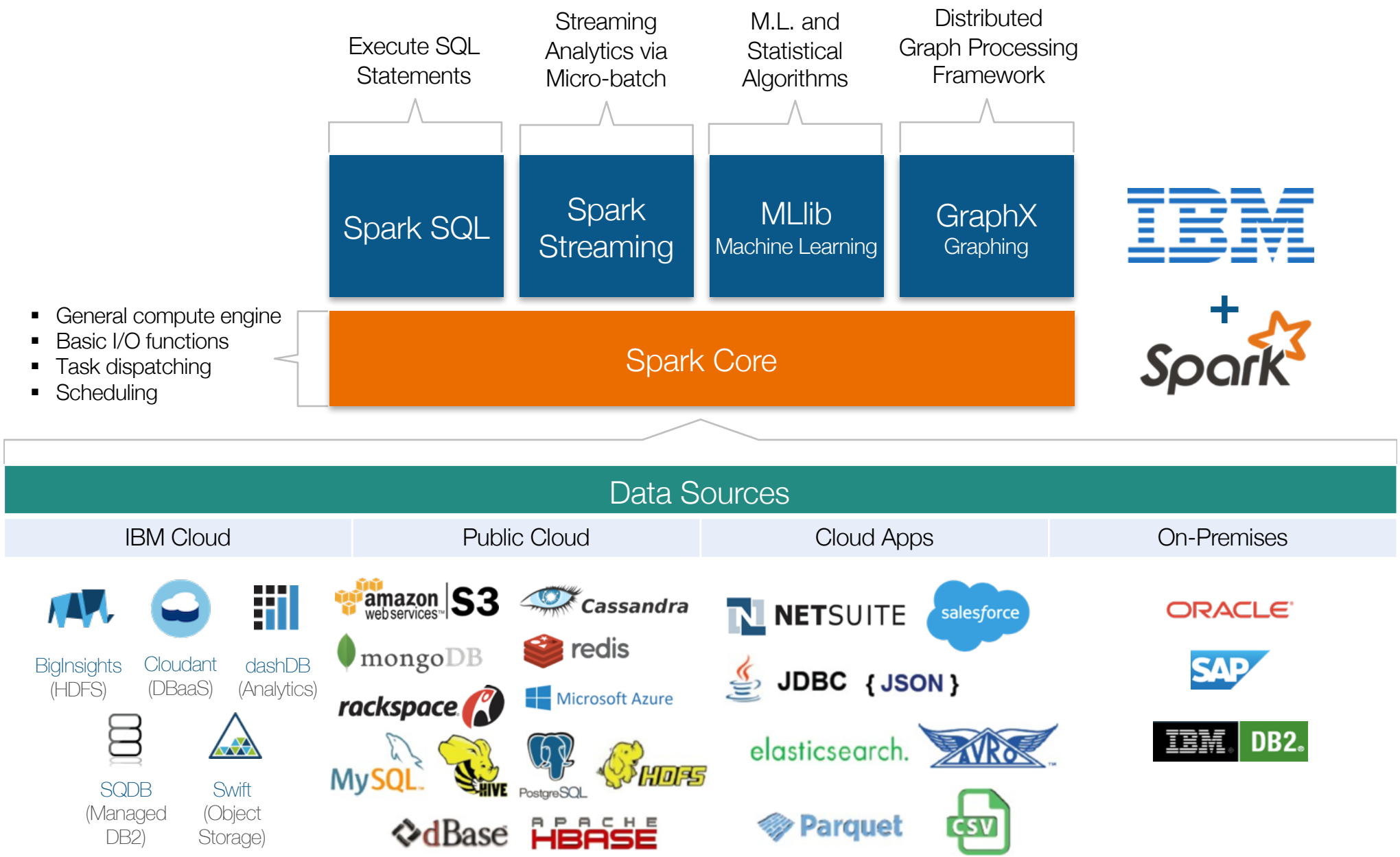


Select Libraries to Meet Use-Case Challenges

- Represent and analyze systems represented by graph nodes
- Trace interconnections between graph nodes
- Applicable to use cases in transportation, telecommunications, road networks, modeling personal relationships, social media, etc.

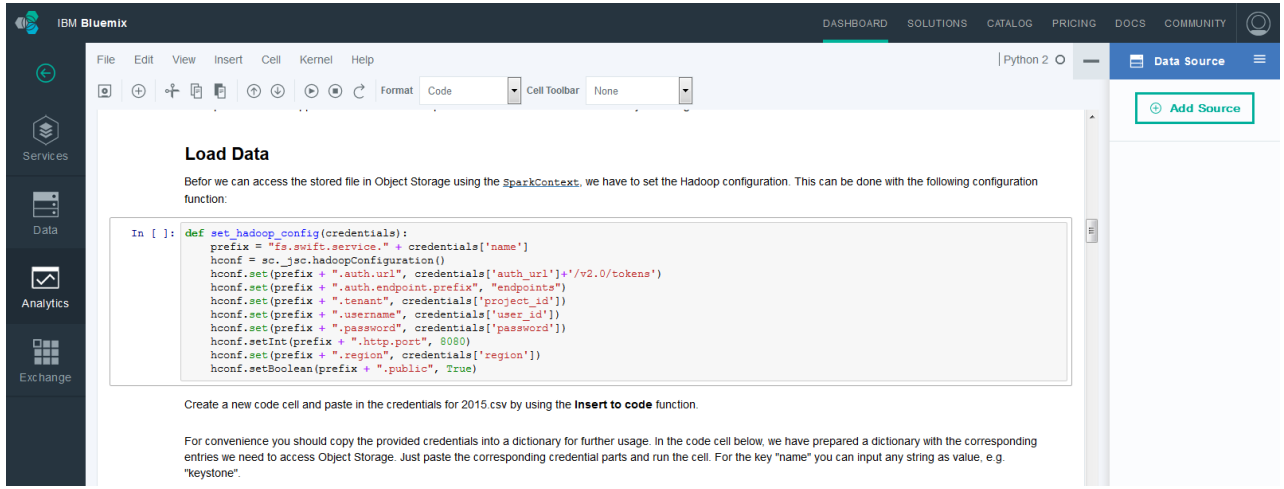


Analytics for Apache Spark – Blends Data Types, Sources, & Workloads



Apache Spark – Notebooks

- **Jupyter Notebooks: Current entry point, integrated with both Personal and Reserved Enterprise editions of IBM Analytics for Apache Spark**
  - Interactive, unified, and collaborative Spark work environments
  - Graphical user interface (GUI) for executing and visualizing the results of Spark programs through Web browsers or consoles



- **Spark-submit API for executing Spark jobs**

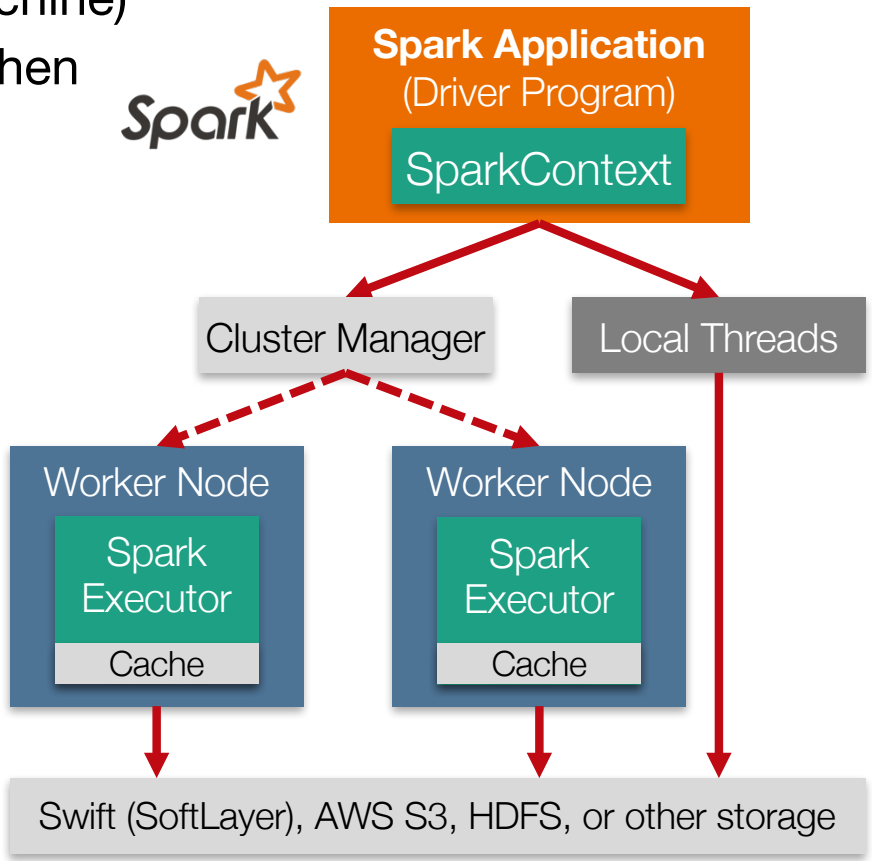
IBM Analytics for Apache Spark

SparkR Programming Tutorial

Spark Drivers & Workers

- **Spark programs generally consist of two components: a driver program and worker program(s)**
  - Worker programs can run on cluster nodes or local threads (if deployed to your local machine)
  - RDDs are created by `SparkContext` and then distributed amongst the workers

- **The `SparkContext` object instructs Spark on how & where to access a cluster**
  - Determines the type and size of cluster available to Spark jobs
  - e.g. Run locally with 1 worker thread VS. Run 'X' local worker threads VS. `spark://HOST:PORT` to a cluster
  - Is created for you on DSX!



Spark Programming Tutorial

In this section, we loosely follow the SparkR tutorial:

http://spark.apache.org/docs/2.0.0/sparkr.html

We also use the SparkR API as a reference:

https://spark.apache.org/docs/2.0.0/api/R/index.html

All the code is available in this notebook:

https://github.com/snowch/demo_21_feb_2017/blob/master/SparkR_Tutorial.ipynb

SparkR – Core concepts: SparkDataFrame

A SparkDataFrame is a **distributed collection** of data organized into **named columns**. It is conceptually equivalent to a table in a relational database or a data frame in R, but with richer optimizations under the hood. SparkDataFrames can be constructed from a wide array of sources such as: structured data files, tables in Hive, external databases, or existing local R data frames.

Creating SparkDataFrames from local data frames

Data Science Experience

My Projects > Scratch > RScratch2

File Edit View Insert Cell Kernel Help

Format Code CellToolbar

```
In [2]: df <- as.data.frame(faithful)
        head(df)
```

eruptions	waiting
3.600	79
1.800	54

API: https://spark.apache.org/docs/2.0.0/api/R/as.data.frame.html

Note that this limits you to the memory available to the Driver program. Later we will see how we can load “Big Data”.

SparkDataFrame Operations

Try these sections from the SparkR guide:

- Selecting rows, columns (API – head, select, filter)
- Grouping, Aggregation (API – summarize, groupBy, arrange)

Homework sections:
(refer to the SparkR API docs as you work through the examples)

- Operations on Columns
- Applying User-Defined Functions
- Run a given function on a large dataset grouping by input column(s) and using gapply or gapplyCollect
- Run local R functions distributed using spark.lapply
- Running SQL Queries from SparkR
- Machine Learning

IBM Analytics for Apache Spark

Working with data in Object Storage

Working with data in Object Storage

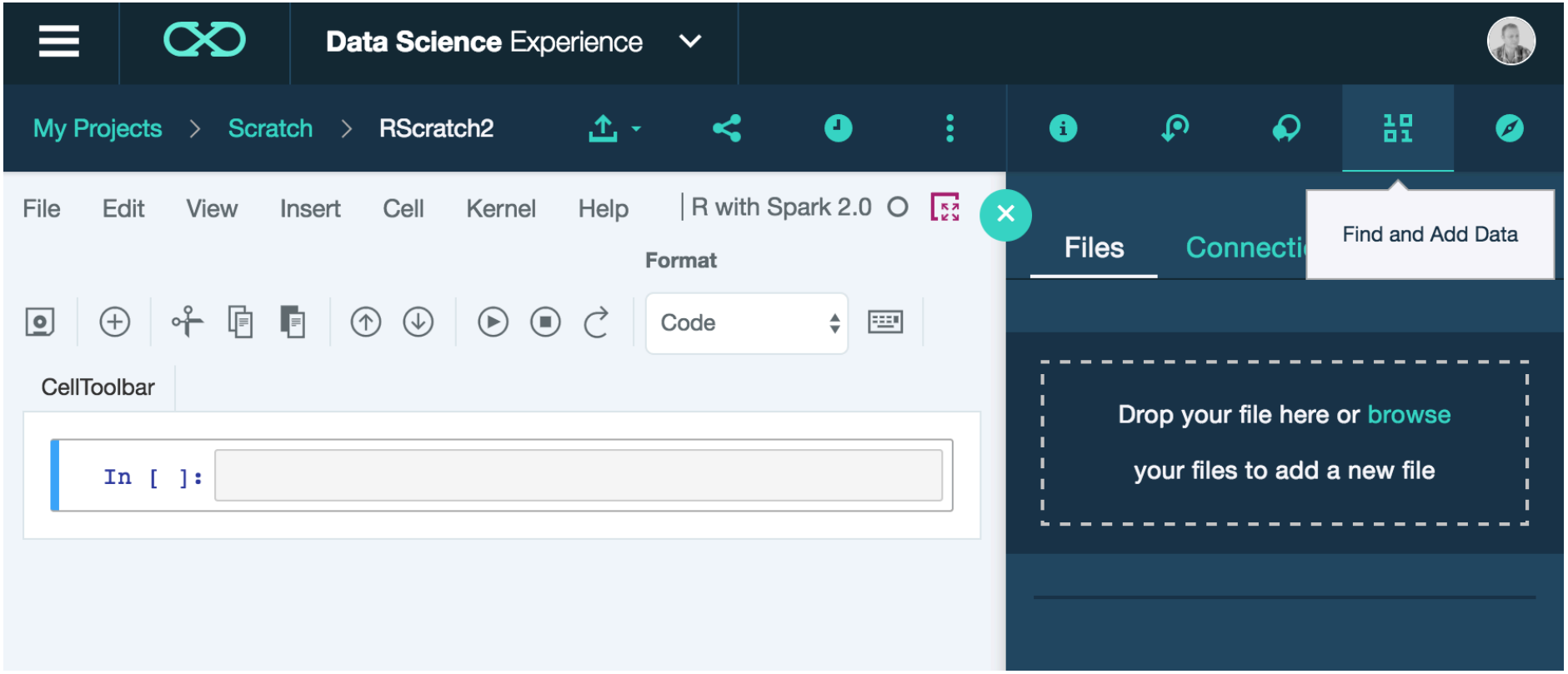
- So far we have created SparkDataFrames from local data frames.
- However, data may be too big to fit just on the Driver node.
- We also need somewhere to store our data. Object store is a popular choice.

- We loosely follow this documentation:

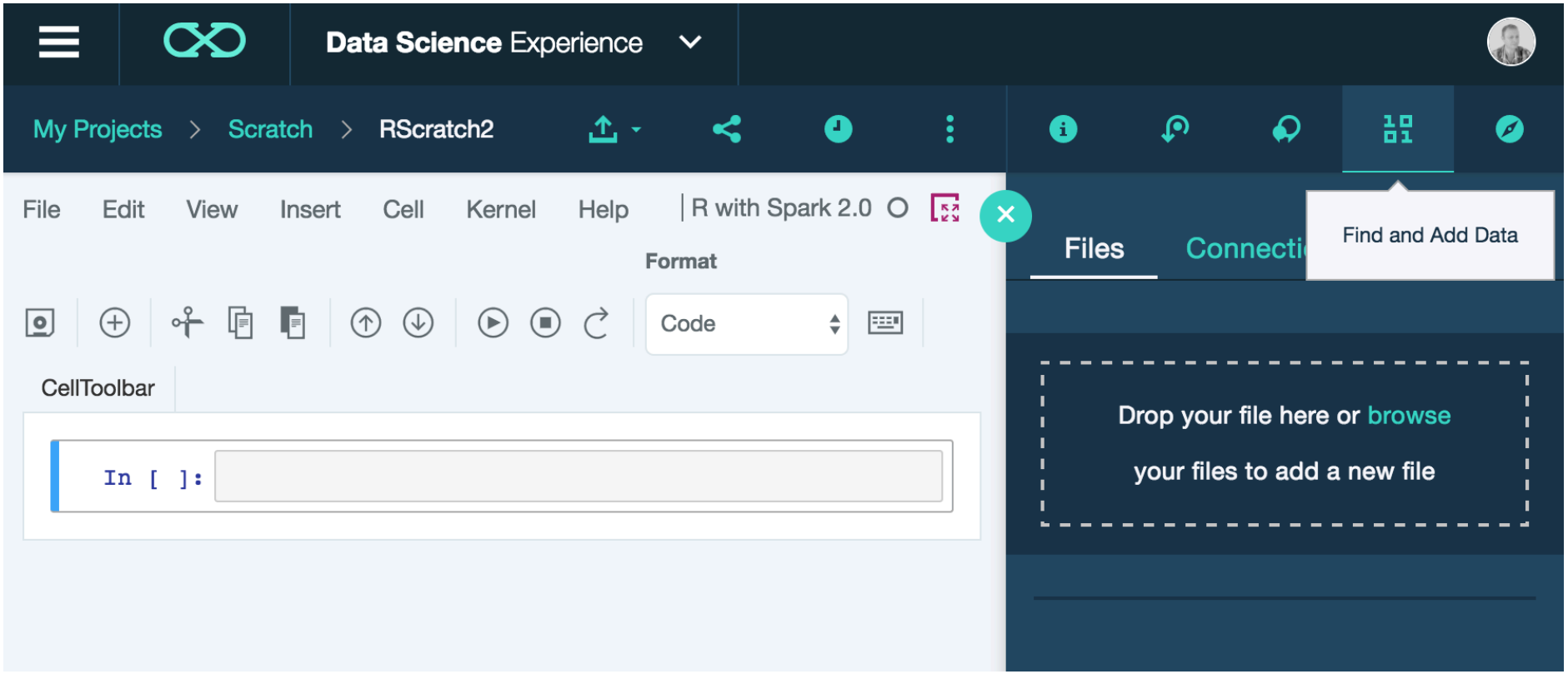
https://apsportal.ibm.com/exchange/public/entry/view/90a34943032a7fde0ced0530d976ca82

Find and Add Data

Drop a file (e.g. CSV) on the palette



Find and Add Data – drop file on the palette

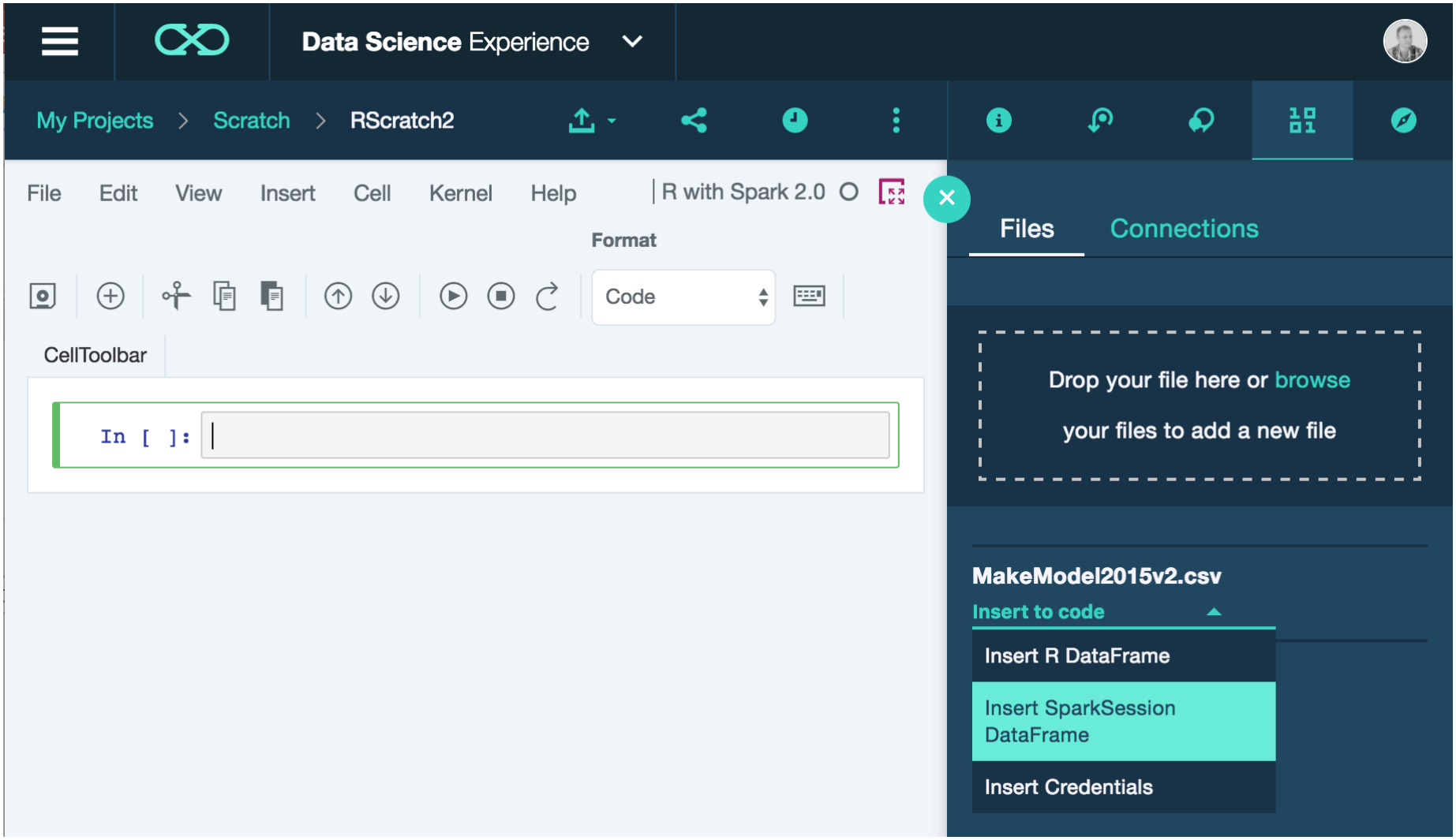


Example dataset:

http://data.dft.gov.uk.s3.amazonaws.com/road-accidents-safety-data/MakeModel2015.zip

Unzip and drop MakeModel2015v2.csv onto the palette to upload it.

Find and Add Data – Insert SparkSession DataFrame



Find and Add Data – You will get some sample code

Code

CellToolbar

In [ ]:

```
# @hidden_cell
# This function is used to setup the access of Spar
# You might want to remove those credentials before
setHadoopConfigWithCredentials_a98886102a7c4ee1b758
    # This function sets the Hadoop configuration s
    # access data from Bluemix Object Storage using

    prefix = paste("fs.swift.service" , name, sep =
    hConf = SparkR:::callJMethod(sc, "hadoopConfigu
    SparkR:::callJMethod(hConf, "set", paste(prefix
    SparkR:::callJMethod(hConf, "set", paste(prefix
    SparkR:::callJMethod(hConf, "set", paste(prefix
    SparkR:::callJMethod(hConf, "set", paste(prefix
    SparkR:::callJMethod(hConf, "set", paste(prefix
    SparkR:::callJMethod(hConf, "set", paste(prefix
    invisible(SparkR:::callJMethod(hConf, "setBoole
}

name <- "keystone"
setHadoopConfigWithCredentials_a98886102a7c4ee1b758

invisible(sparkR.session(appName = "test SparkSessi

df.data.1 <- read.df(paste("swift://", "Scratch", "
head(df.data.1)
```

Drop your file here or browse

your files to add a new file

MakeModel2015v2.csv

Insert to code

Find and Add Data – Tweak the example code

Locate this line:

```
df.data.1 <- read.df(paste("swift://", "Scratch", "." , name,"/",
            "MakeModel2015v2.csv", sep=""), source =
            "org.apache.spark.sql.execution.datasources.csv.CSVFileFormat",
            header = ”true")
```

Change

header = “true” to header = “false”

Then execute the cell.

Find and Add Data – You should now see some data

Data Science Experience

My Projects > Scratch > RScratch2

File Edit View Insert Cell Kernel Help | R with Spark 2.0

Format Code CellToolbar

```
invisible(sparkR.session(appName = "test SparkSession R"))

df.data.1 <- read.df(paste("swift://", "Scratch", "." , name,"/", "MakeModel2015v2.csv", sep=""), source = "or
head(df.data.1)
```

_c0	_c1	_c2	_c3	_c4	_c5	_c6	_c7	_c8	_c9	...	_c14	_c15	_c16	_c17	_c18	_c19	_c20	_c21	_c22	_c23
201501BS70001	2015	1	19	0	9	0	8	0	0	...	1	1	-1	2143	2	4	-1	-1	MERCEDES-BENZ	SPRINTEI 310 CDI
201501BS70002	2015	1	9	0	9	0	8	0	0	...	6	1	-1	1600	1	3	-1	-1	BMW	118I SE TURBO AUTO
201501BS70004	2015	1	9	0	9	0	2	0	0	...	6	1	6	1686	2	10	-1	1	VAUXHALL	ASTRA CLUB CD
201501BS70009	2015	1	3	0	18	0	8	0	0	...	6	1	7	124	1	3	-1	1	YAMAHA	NXC 125 CYGNUS
201501BS70009	2015	2	19	0	6	0	8	0	0	...	1	1	7	2402	2	5	-1	1	FORD	TRANSIT 115 T330: RWD
201501BS70010	2015	1	9	0	9	0	8	0	0	...	6	1	7	1461	2	10	-1	1	RENAULT	MEGANE DYNAMIC DCI 80

In [ ]:

Summary

- You have now followed a process that can scale for handling Big Data
- With the free DSX tier you have 5GB storage and 2 Spark Executors
- This can be expanded via paid plans.
- See the pricing on Bluemix for more information:
  - https://console.ng.bluemix.net/catalog/services/apache-spark/

Apache Spark
Technical Enablement

For questions about this presentation, please contact

Chris Snow

Big Data Architect – IBM Analytics

chris.snow@uk.ibm.com