

Apache Spark

Crash Course - DataWorks Summit – Berlin 2018

Robert Hryniewicz

Data Evangelist

@RobHryniewicz



Data

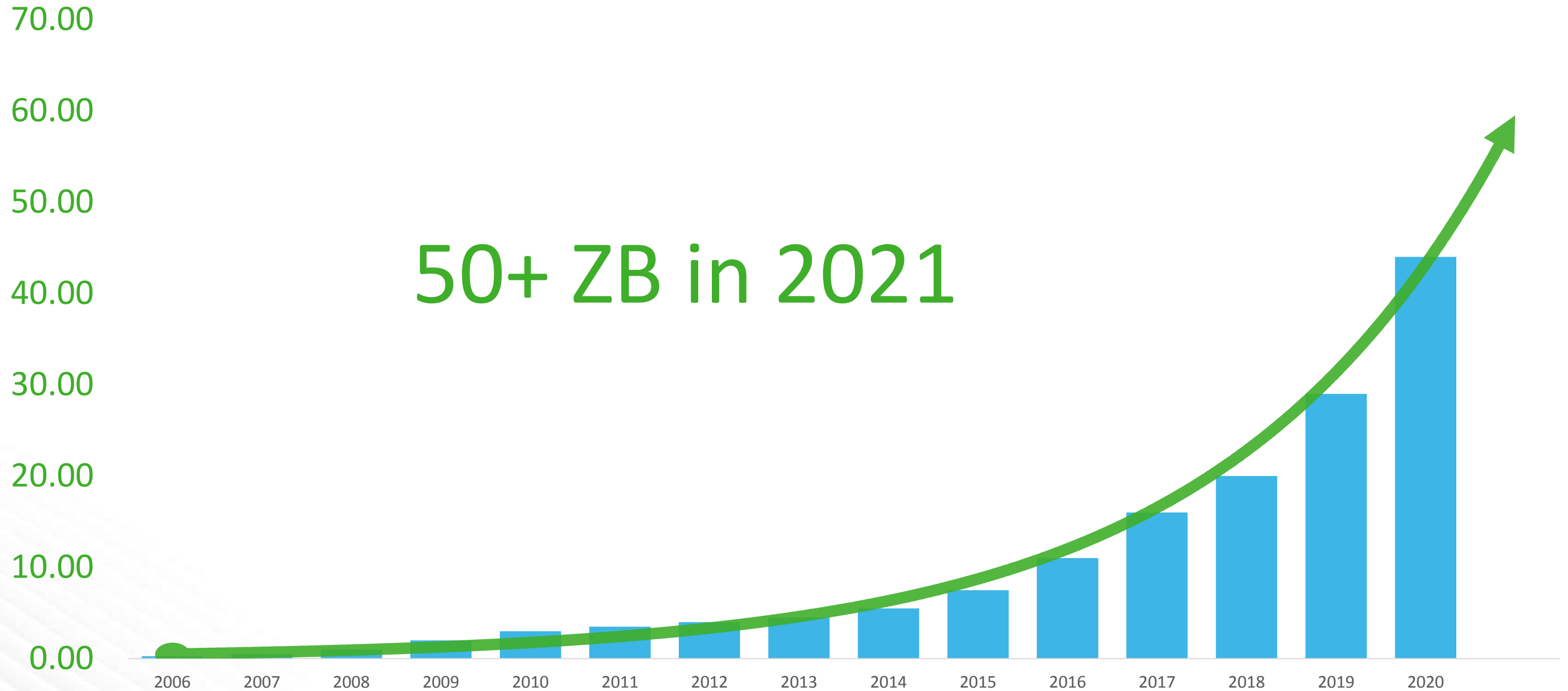
Data Sources

- ◆ Internet of Things (IoT)
 - Wind Turbines, Oil Rigs
 - Beacons, Wearables
 - Smart Cars
- ◆ User Generated Content (Social, Web & Mobile)
 - Twitter, Facebook, Snapchat
 - Clickstream
 - Paypal, Venmo



Data Growth in Zeta Bytes (ZB)

50+ ZB in 2021



The “Big Data” Problem

Problem

- ◆ A single machine cannot process or even store all the data!

Solution

- ◆ Distribute data over large clusters

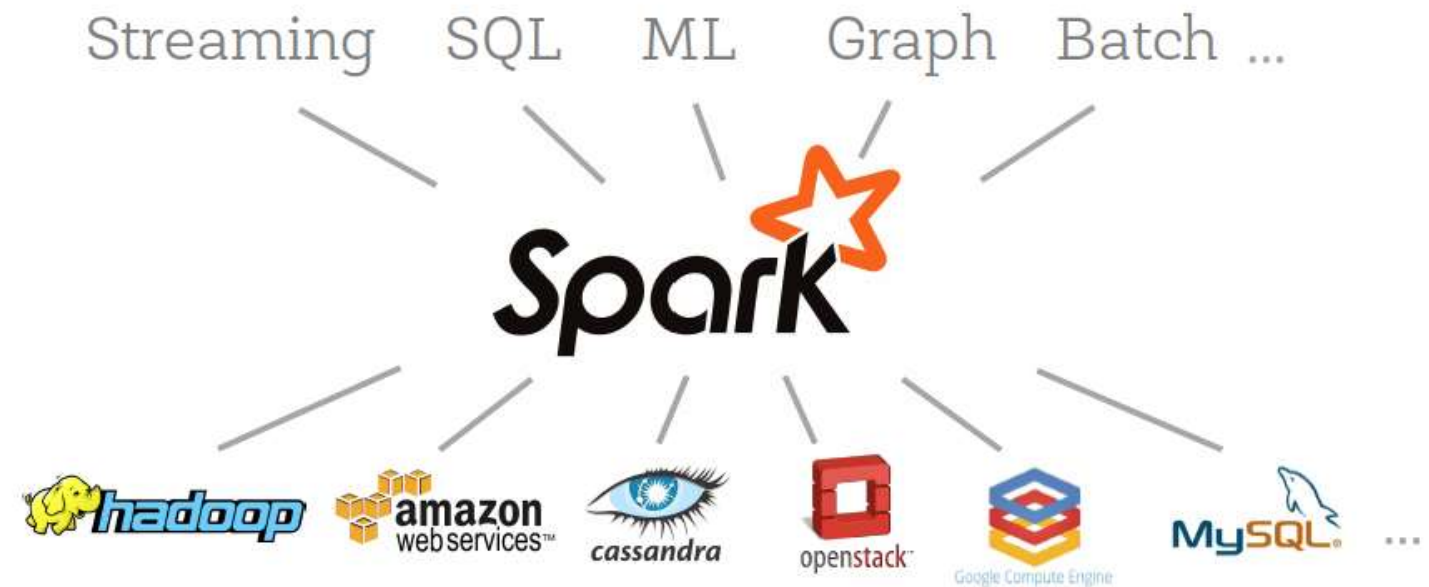
Difficulty

- ◆ How to split work across machines?
- ◆ Moving data over network is expensive
- ◆ Must consider data & network locality
- ◆ How to deal with failures?
- ◆ How to deal with slow nodes?

Apache Spark

What Is Apache Spark?

- ◆ **Apache open source project** originally developed at AMPLab (University of California Berkeley)
- ◆ **Unified, general data processing engine** that operates across varied data workloads and platforms



Why Apache Spark?

◆ **Elegant Developer APIs**

- Single environment for data munging, data wrangling, and Machine Learning (ML)

◆ **In-memory computation model – Fast!**

- Effective for iterative computations and ML

◆ **Machine Learning**

- Implementation of distributed ML algorithms
- Pipeline API (Spark MLlib)
- External libraries via open & commercial projects (H2Os Sparkling Water)

Spark SQL

Structured Data

Spark Streaming

Real-time

Spark MLlib

Machine Learning

GraphX

Graph Analysis



Spark SQL

Spark SQL

Structured Data

Spark Streaming

Near Real-time

Spark MLlib

Machine Learning

GraphX

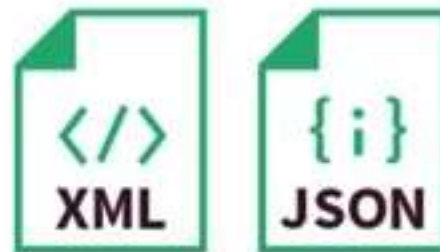
Graph Analysis



UNSTRUCTURED



SEMI-STRUCTURED



STRUCTURED



More Flexible

///

Better Storage and Performance

Spark SQL Overview

- ◆ Spark module for *structured data* processing (e.g. ORC, Parquet, Avro, MySQL)
- ◆ Two ways to manipulate data:
 - DataFrame/Dataset API
 - SQL query

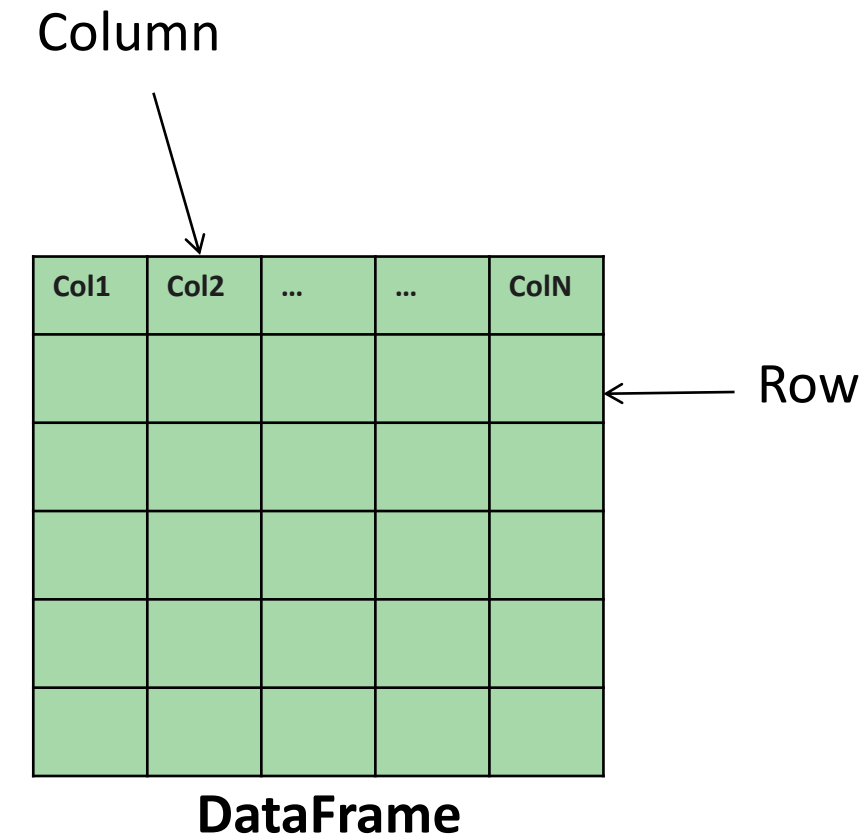
SparkSession

What is it?

- ◆ Main entry point for Spark functionality
- ◆ Allows programming with DataFrame and Dataset APIs
- ◆ Represented as **spark** and auto-initialized in a notebook type env. (Zeppelin or Jupyter)

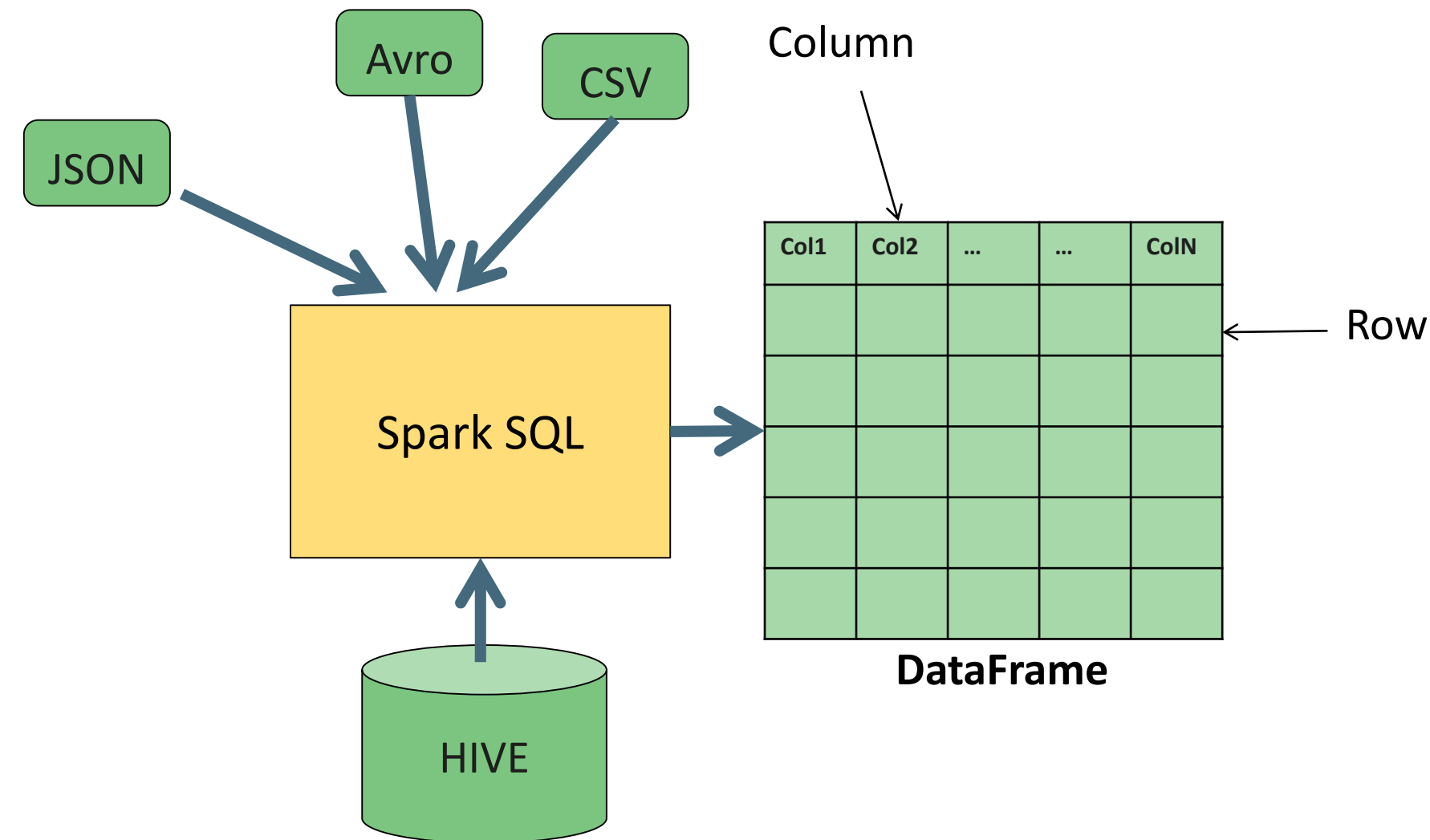
DataFrames

- ◆ Distributed collection of data organized into *named columns*
- ◆ Conceptually equivalent to a table in **relational DB** or a **data frame in R/Python**
- ◆ API available in Scala, Java, Python, and R



Data is described as a DataFrame with **rows**, **columns**, and a **schema**

Sources



Create a DataFrame

Example

```
val path = "examples/flights.json"  
val flights = spark.read.json(path)
```


Register a Temporary View (SQL API)

Example

```
flights.createOrReplaceTempView("flightsView")
```

Two API Examples: DataFrame and SQL APIs

DataFrame API

```
flights.select("Origin", "Dest", "DepDelay")  
        .filter($"DepDelay" > 15).show(5)
```

SQL API

```
SELECT Origin, Dest, DepDelay  
FROM flightsView  
WHERE DepDelay > 15 LIMIT 5
```

Results

| Origin | Dest | DepDelay |
|--------|------|----------|
| IAD | TPA | 19 |
| IND | BWI | 34 |
| IND | JAX | 25 |
| IND | LAS | 67 |
| IND | MCO | 94 |

Spark Streaming

Spark SQL

Structured Data

Spark Streaming

Real-time

Spark MLlib

Machine Learning

GraphX

Graph Analysis



What is Stream Processing?

Batch Processing

- Ability to process and analyze data at-rest (stored data)
- Request-based, bulk evaluation and short-lived processing
- Enabler for **Retrospective, Reactive and On-demand Analytics**

Stream Processing

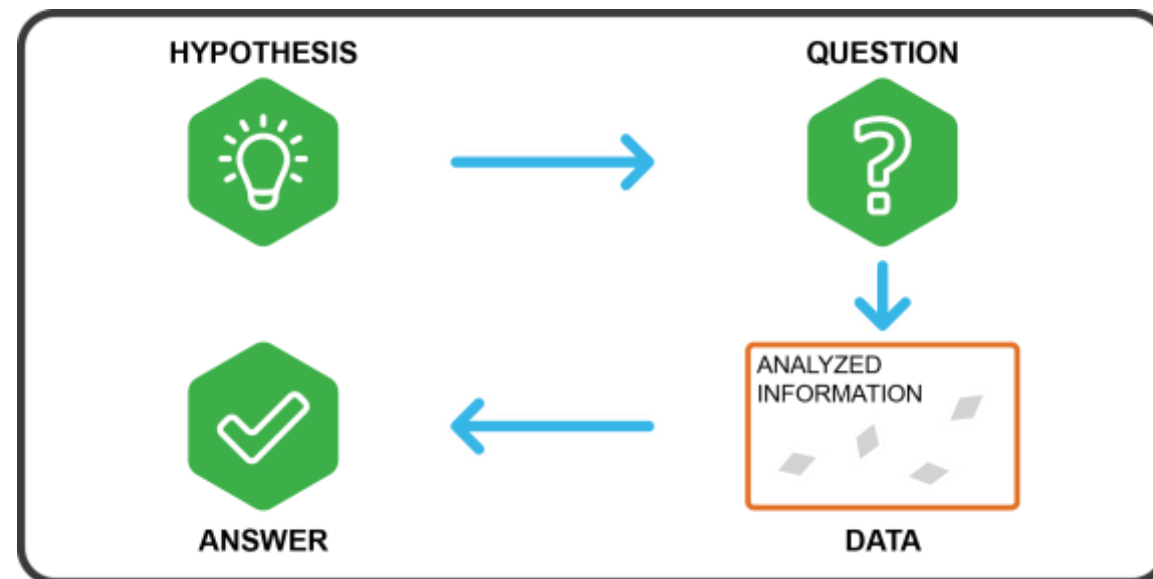
- Ability to ingest, process and analyze data in-motion in real- or near-real-time
- Event or micro-batch driven, continuous evaluation and long-lived processing
- Enabler for **real-time Prospective, Proactive and Predictive Analytics** for Next Best Action

Stream Processing + **Batch Processing** = **All Data Analytics**
real-time (now) historical (past)

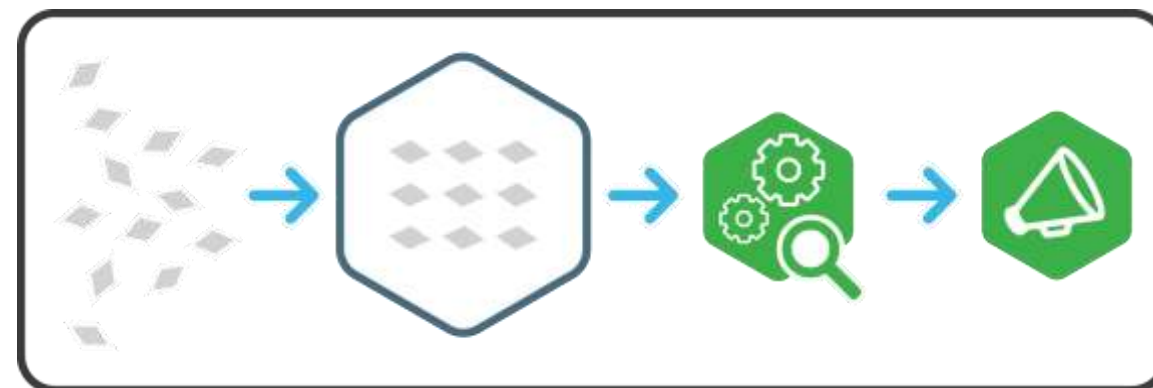
Modern Data Applications approach to Insights

Traditional Analytics

Structured & Repeatable
Structure built to store data



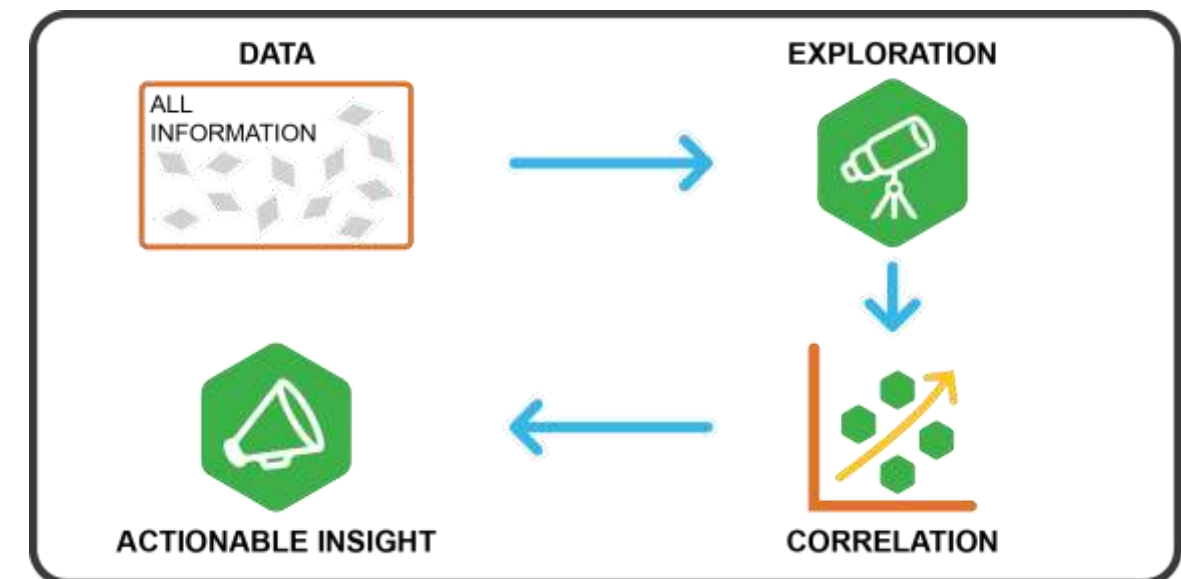
Start with hypothesis
Test against selected data



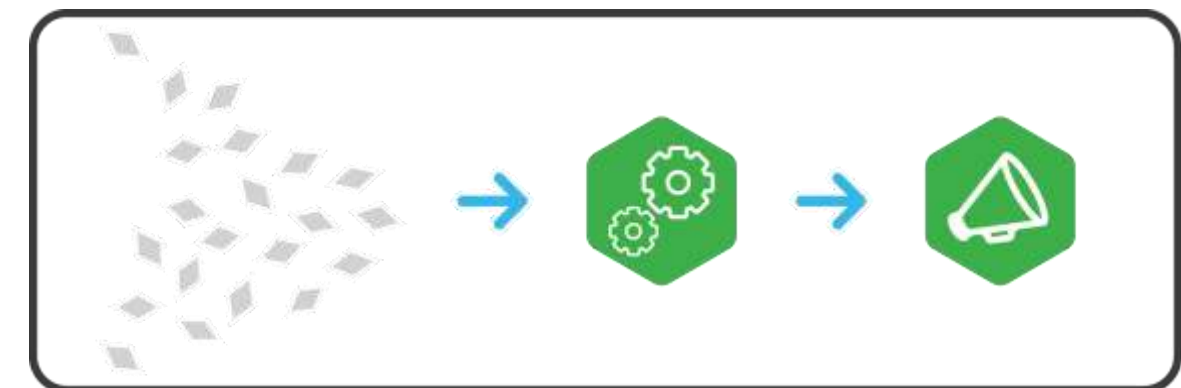
Analyze after landing...

Next Generation Analytics

Iterative & Exploratory
Data is the structure



Data leads the way
Explore *all* data, identify correlations



Analyze in motion...

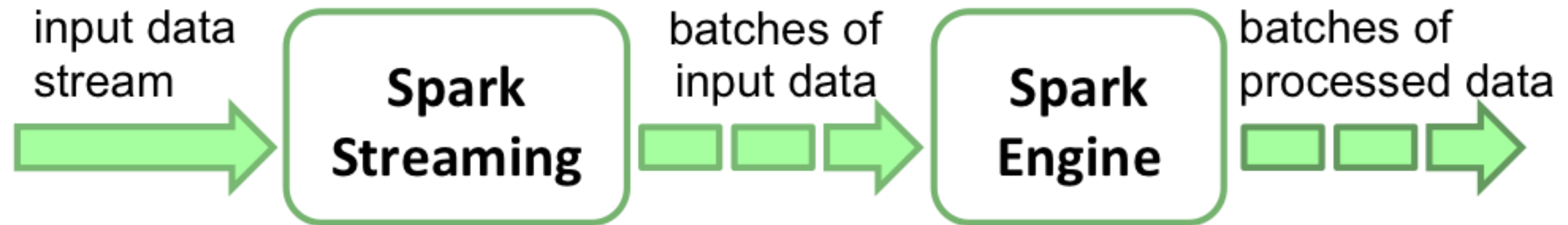
Spark Streaming

Overview

- ◆ Extension of Spark Core API
- ◆ Stream processing of live data streams
 - Scalable
 - High-throughput
 - Fault-tolerant



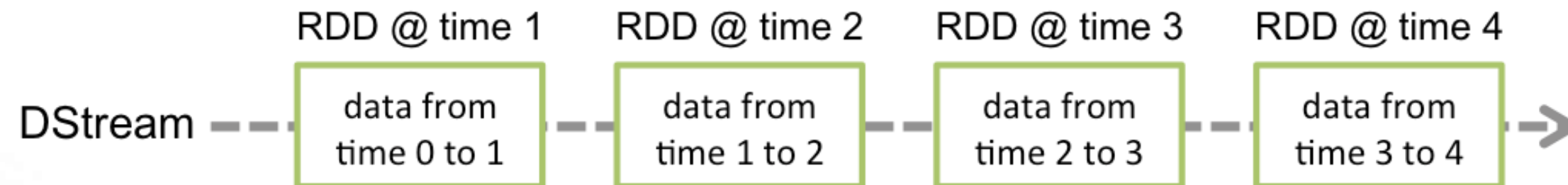
Spark Streaming



Spark Streaming

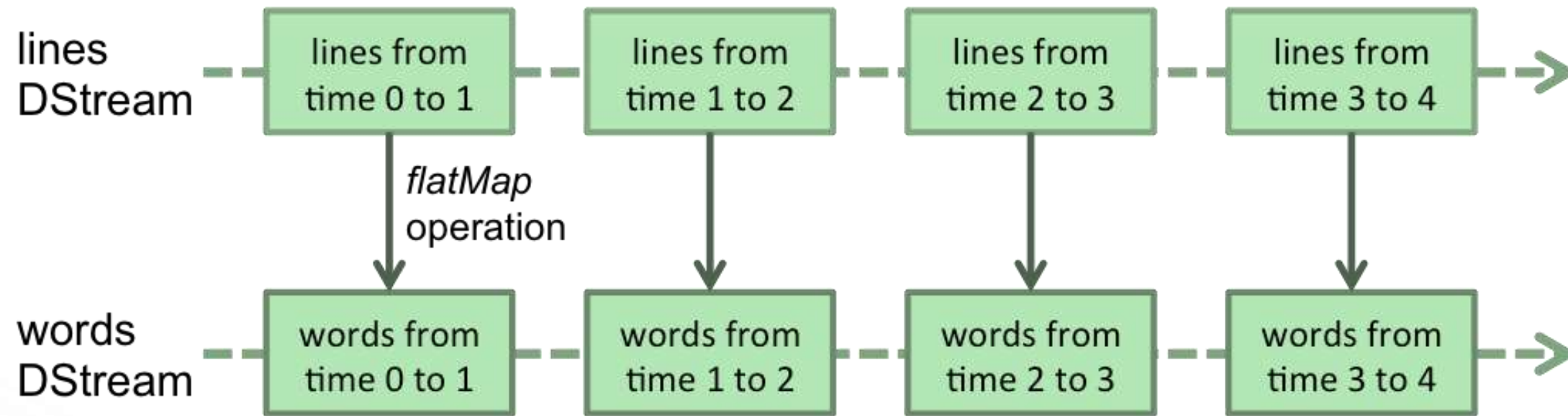
Discretized Streams (DStreams)

- ◆ High-level abstraction representing **continuous stream of data**
- ◆ Internally represented as a sequence of RDDs
- ◆ Operation applied on a DStream translates to operations on the underlying RDDs



Spark Streaming

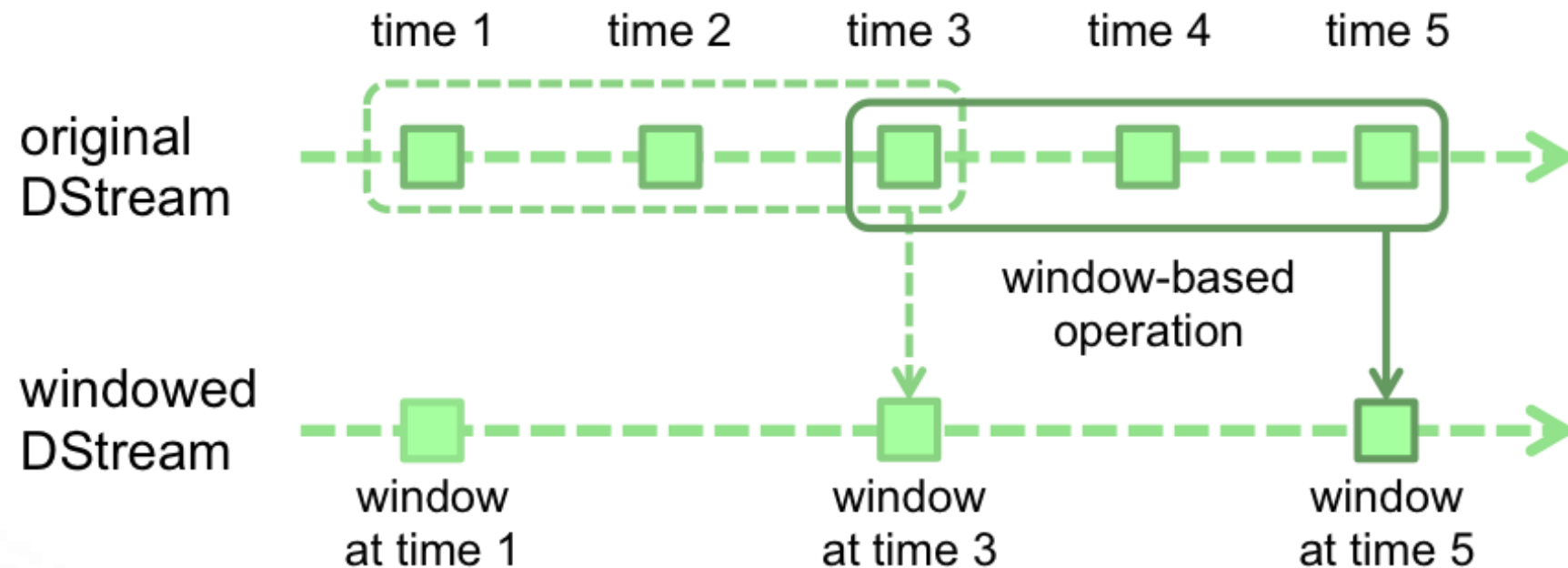
Example: *flatMap* operation



Spark Streaming

Window Operations

- ◆ Apply transformations over a sliding window of data, e.g. rolling average



```
// Reduce last 30 seconds of data, every 10 seconds
```

```
val windowedWordCounts = pairs.reduceByKeyAndWindow((a:Int,b:Int) => (a + b), Seconds(30), Seconds(10))
```

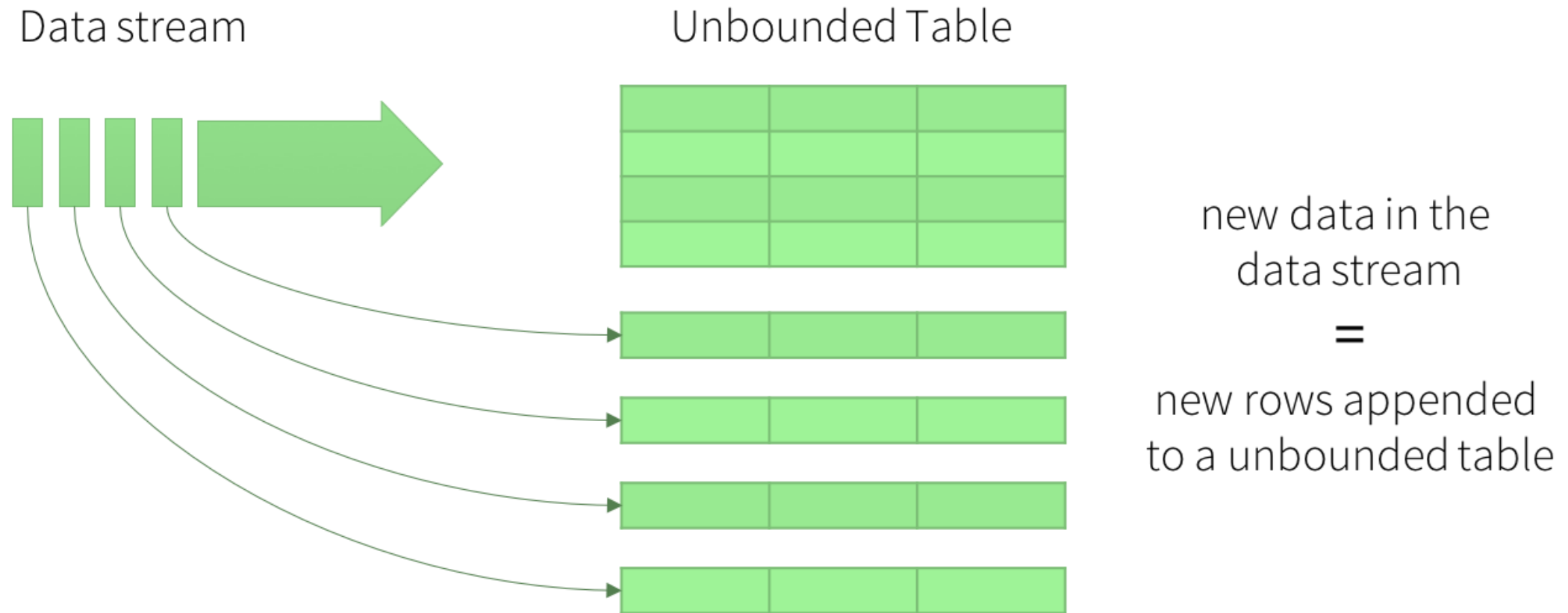
Challenges in Streaming Data

- ◆ Consistency
- ◆ Fault tolerance
- ◆ Out-of-order data

Structured Streaming

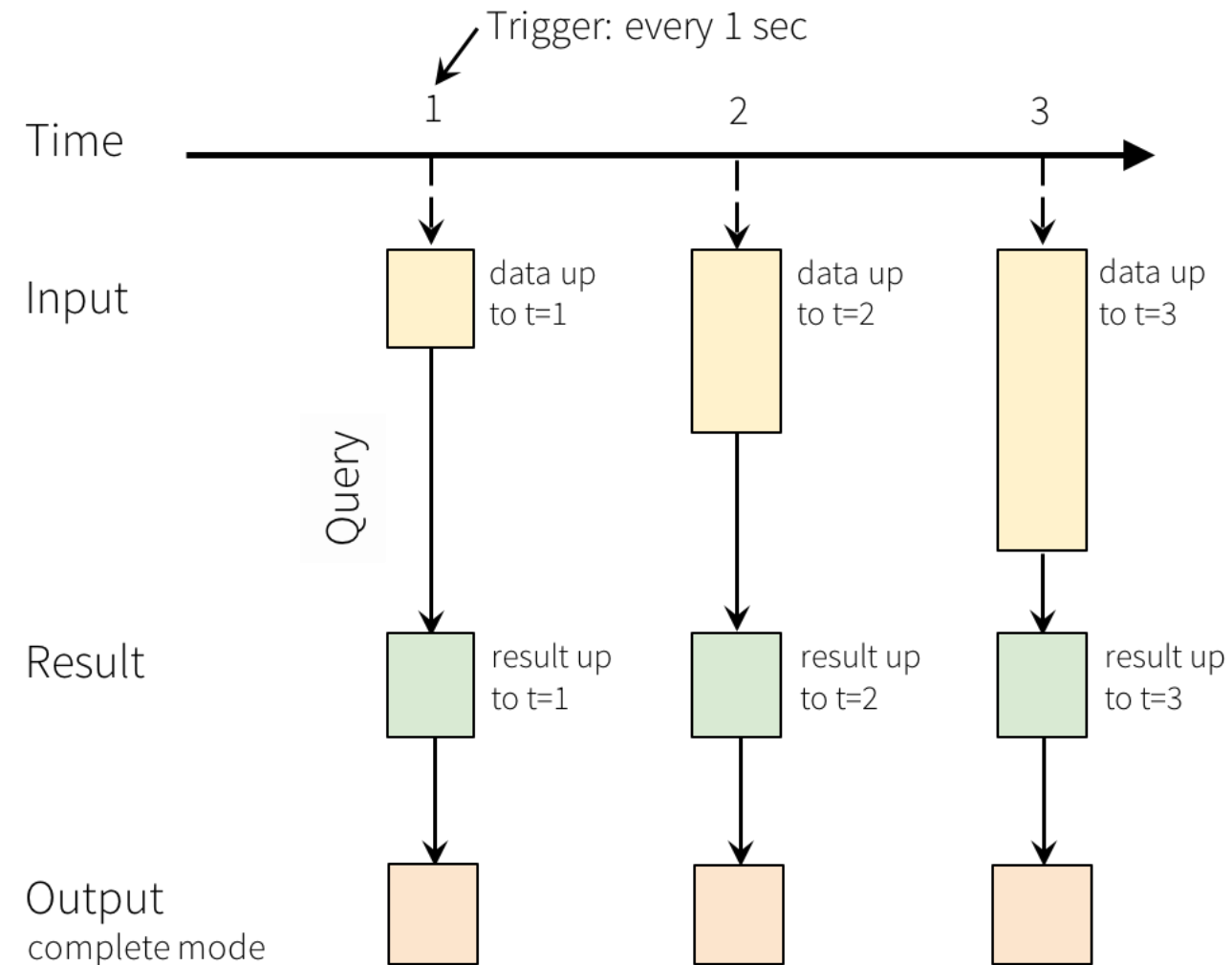
- ◆ High-Level APIs - DataFrames, Datasets and SQL. Same in streaming and in batch
- ◆ Event-time Processing - Native support for working w/ out-of-order and late data
- ◆ End-to-end Exactly Once - Transactional both in processing and output

Structured Streaming: Basics



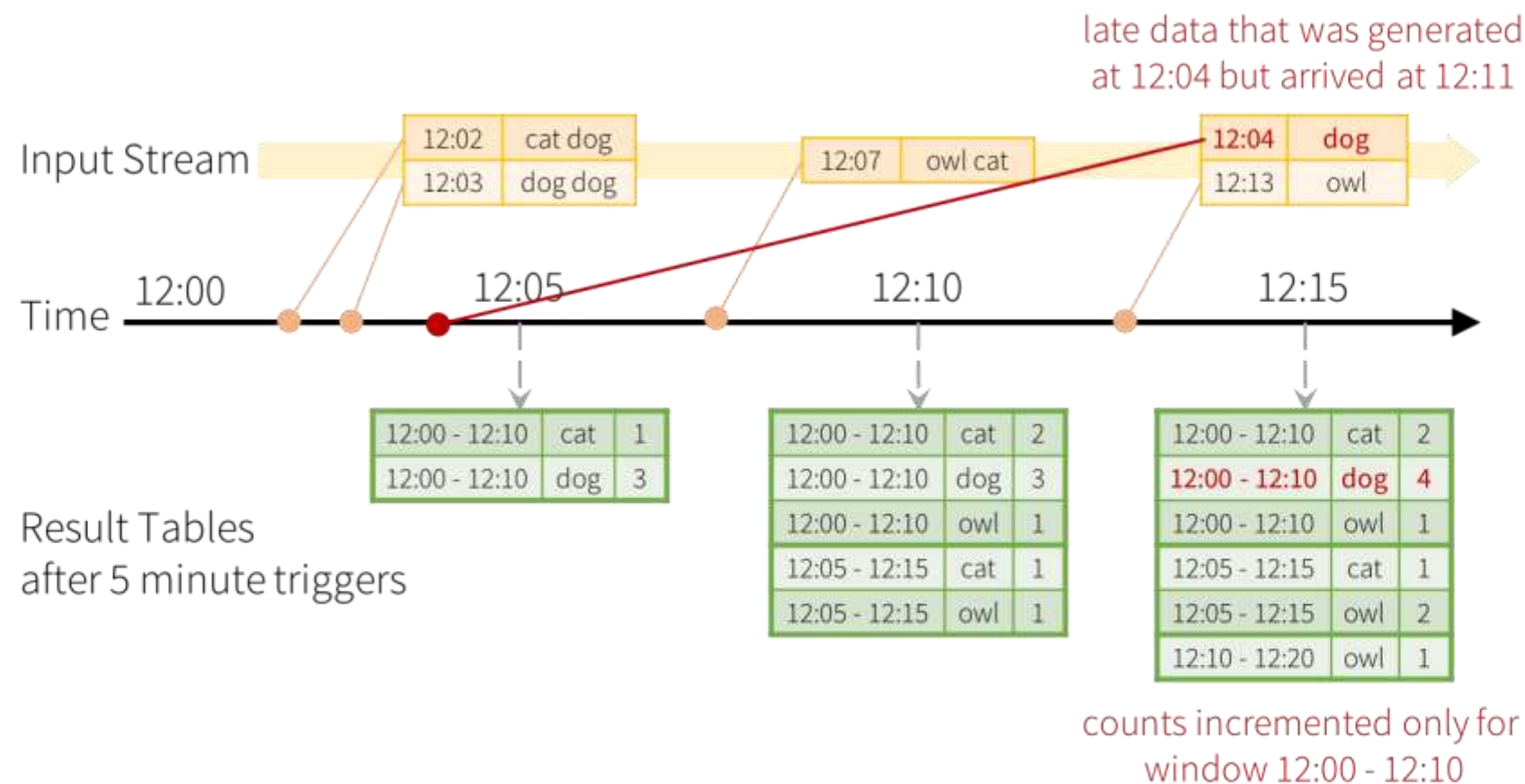
Data stream as an unbounded table

Structured Streaming: Model



Programming Model for Structured Streaming

Handling late arriving data



Late data handling in
Windowed Grouped Aggregation

Spark MLlib

Spark SQL

Structured Data

Spark Streaming

Near Real-time

Spark MLlib

Machine Learning

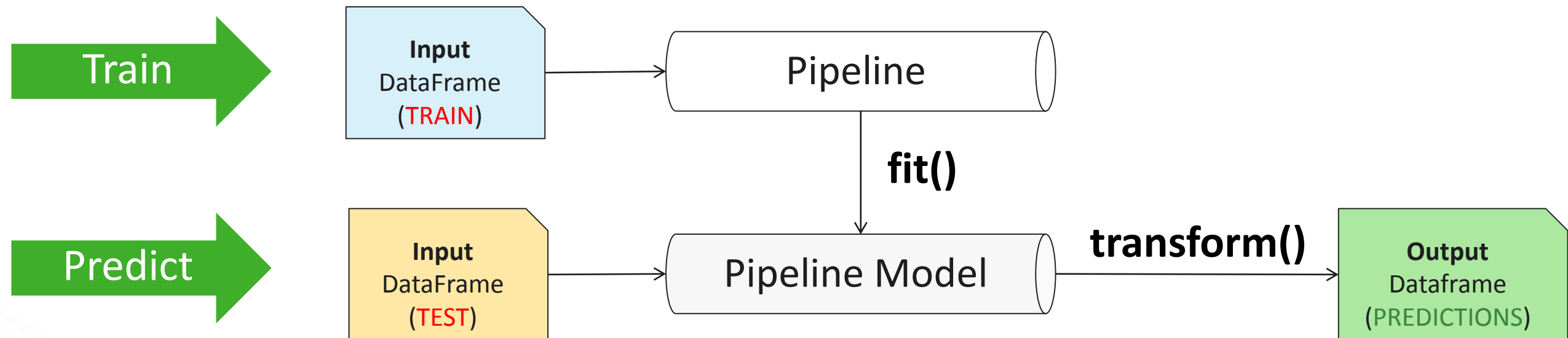
GraphX

Graph Analysis

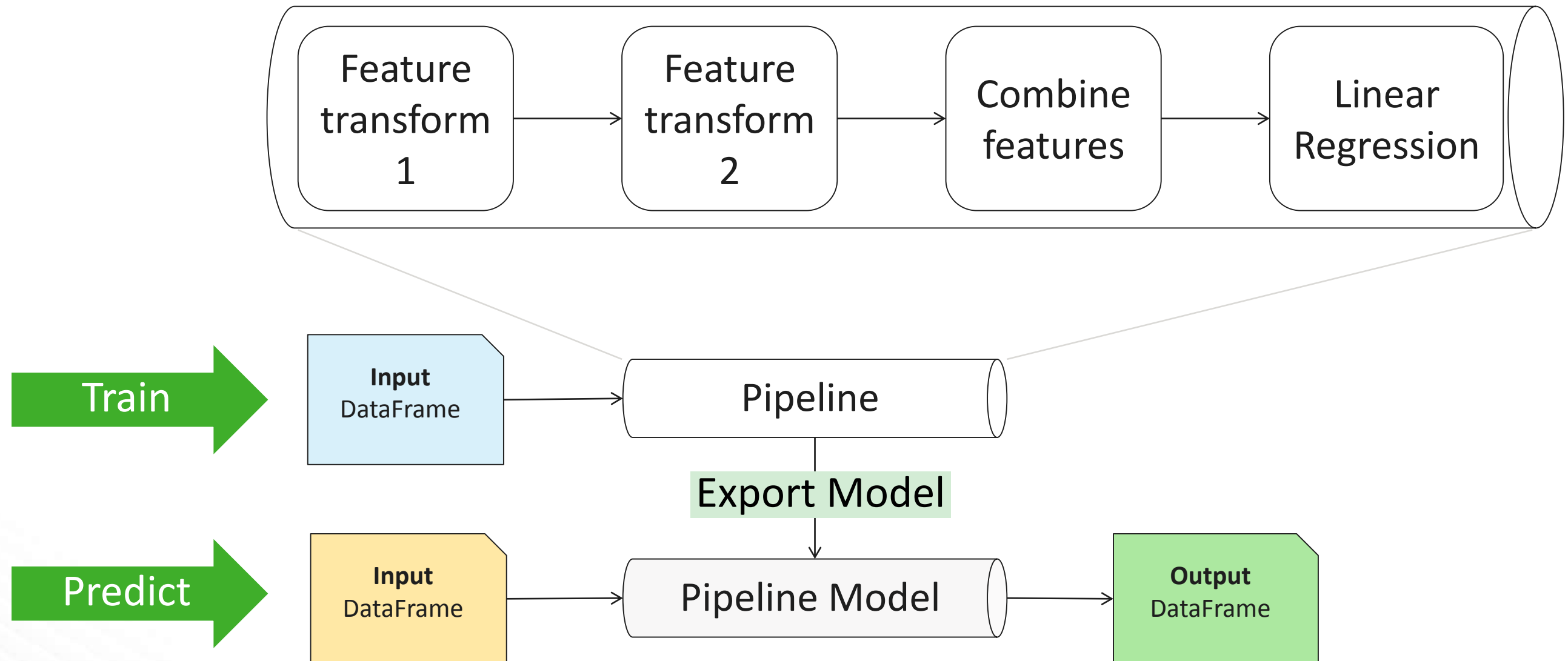


Spark ML Pipeline

- ◆ **fit()** is for **training**
- ◆ **transform()** is for **prediction**



Spark ML Pipeline



Sample Spark ML Pipeline

```
indexer = ...
parser = ...
hashingTF = ...
vecAssembler = ...

rf = RandomForestClassifier(numTrees=100)
pipe = Pipeline(stages=[indexer, parser, hashingTF, vecAssembler, rf])

model = pipe.fit(trainData)           # Train model
results = model.transform(testData)    # Test model
```

Exporting ML Models - PMML

- ◆ Predictive Model Markup Language (PMML)
 - > XML-based predictive model interchange format
- ◆ Supported models
 - K-Means
 - Linear Regression
 - Ridge Regression
 - Lasso
 - SVM
 - Binary

Spark GraphX

Spark SQL

Structured Data

Spark Streaming

Near Real-time

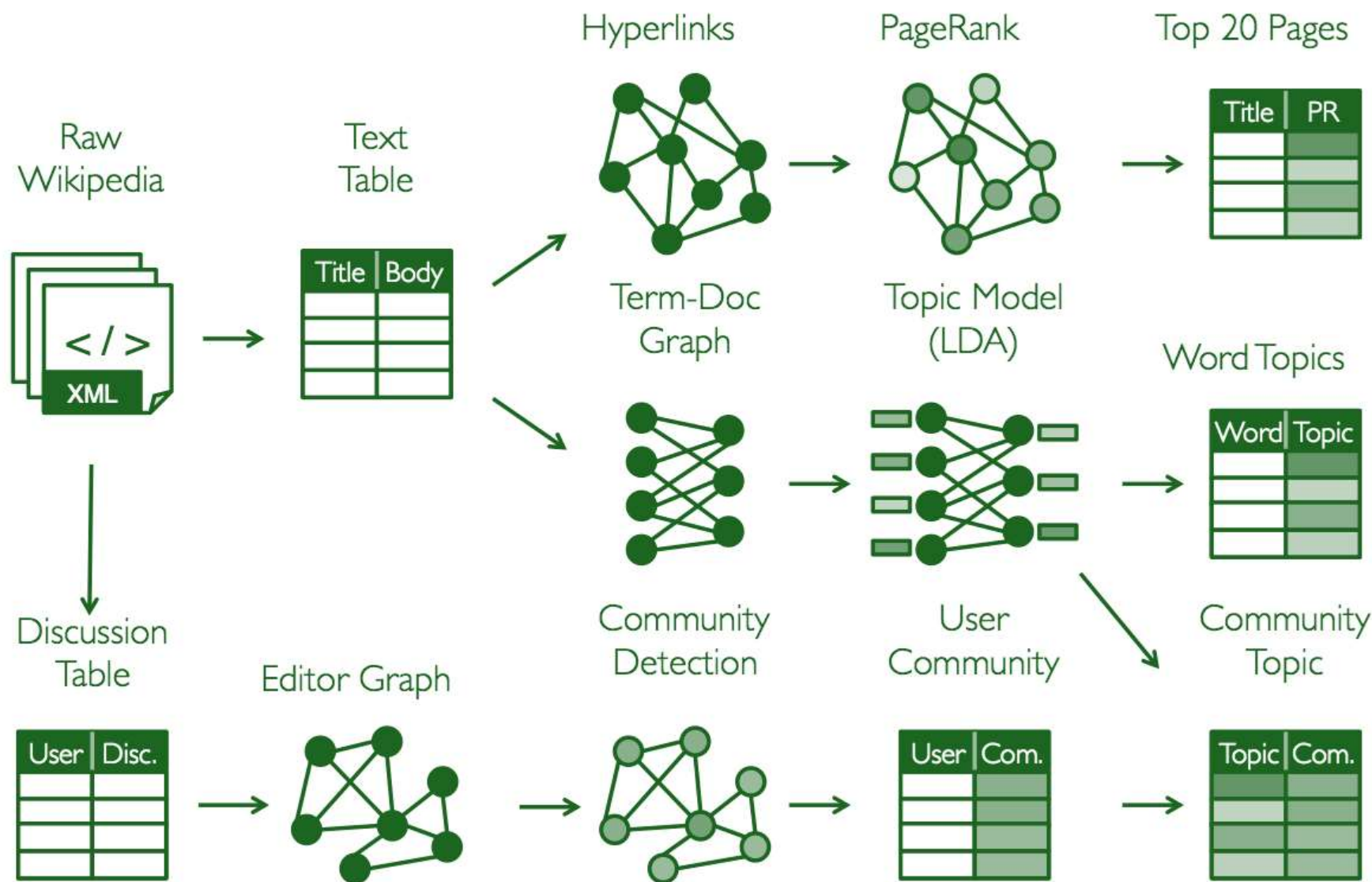
Spark MLlib

Machine Learning

GraphX

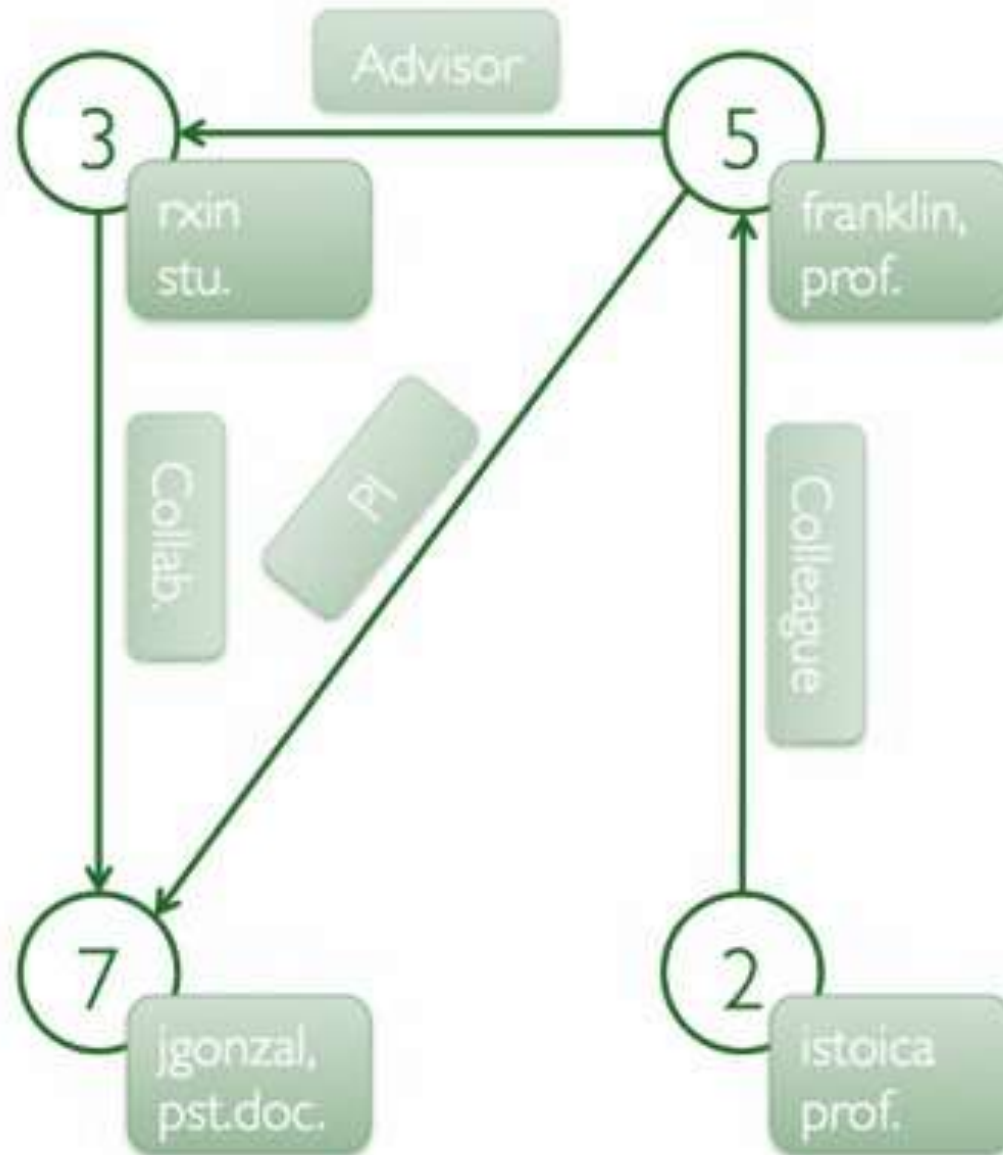
Graph Analysis





- Page Rank
- Topic Modeling (LDA)
- Community Detection

Property Graph



Vertex Table

| Id | Property (V) |
|----|-----------------------|
| 3 | (rxin, student) |
| 7 | (jgonzal, postdoc) |
| 5 | (franklin, professor) |
| 2 | (istoica, professor) |

Edge Table

| SrclId | DstId | Property (E) |
|--------|-------|--------------|
| 3 | 7 | Collaborator |
| 5 | 3 | Advisor |
| 2 | 5 | Colleague |
| 5 | 7 | PI |

GraphX Algorithms

- ◆ PageRank
- ◆ Connected components
- ◆ Label propagation
- ◆ SVD++
- ◆ Strongly connected components
- ◆ Triangle count

Sample GraphX Code in Scala

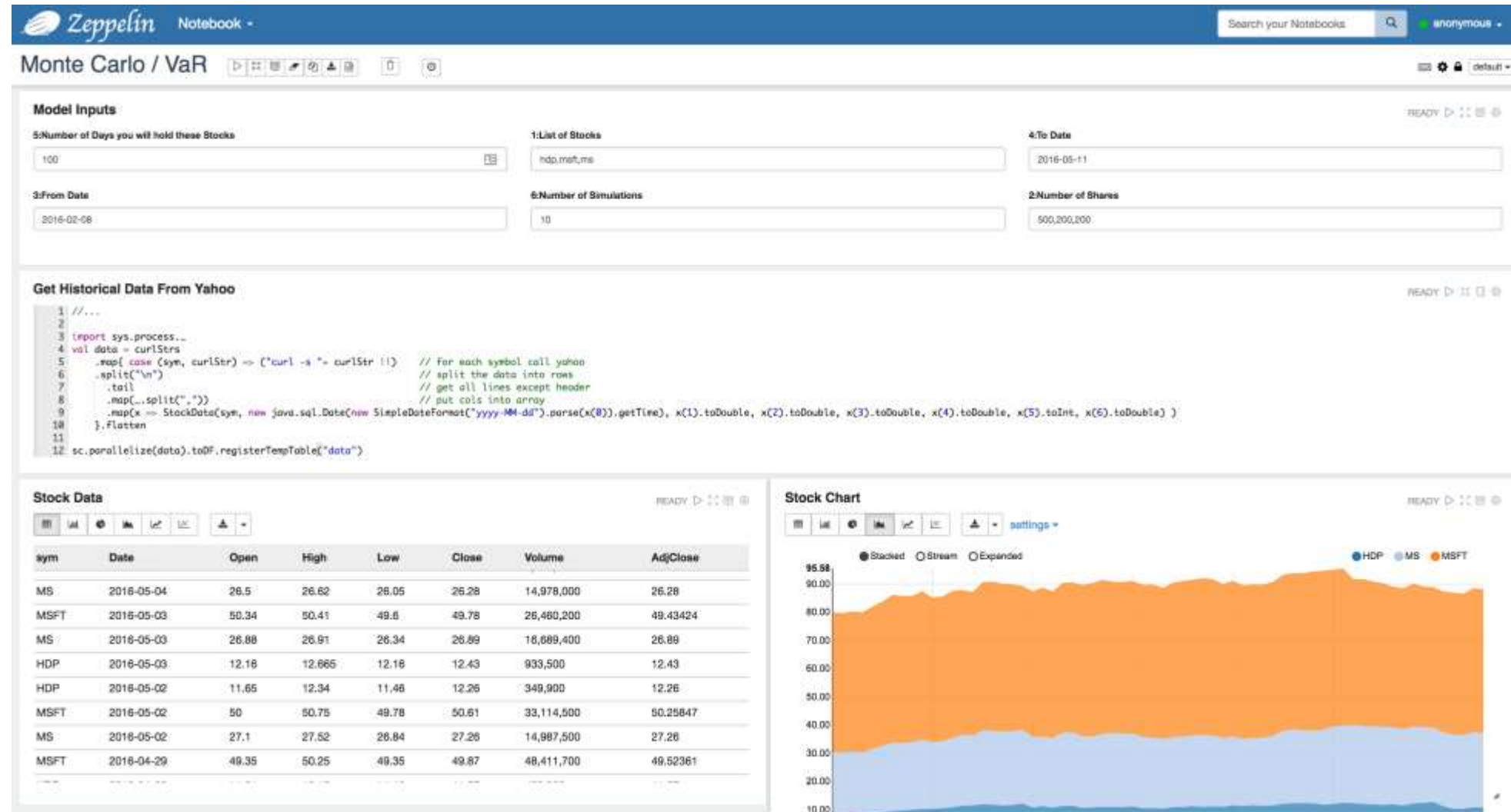
```
graph = Graph(vertices, edges)
messages = spark.textFile("hdfs://...")
graph2 = graph.joinVertices(messages) {
  (id, vertex, msg) => ...
}
```


Apache Zeppelin

What's Apache Zeppelin?

Web-based notebook that enables interactive data analytics.

You can make beautiful data-driven, interactive and collaborative documents with SQL, Python, Scala and more



Apache Zeppelin with HDP 2.6+

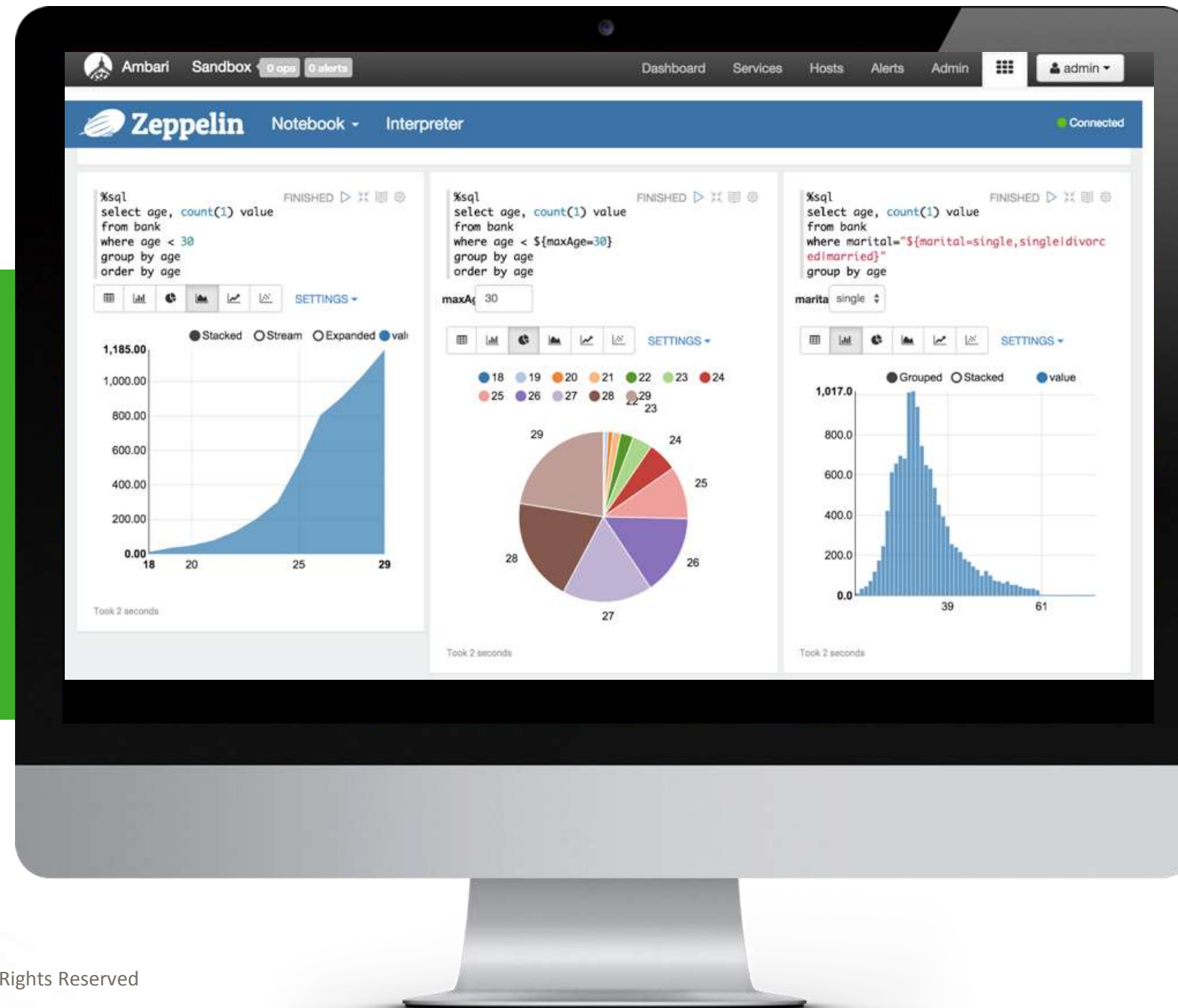
Web-based Notebook for interactive analytics

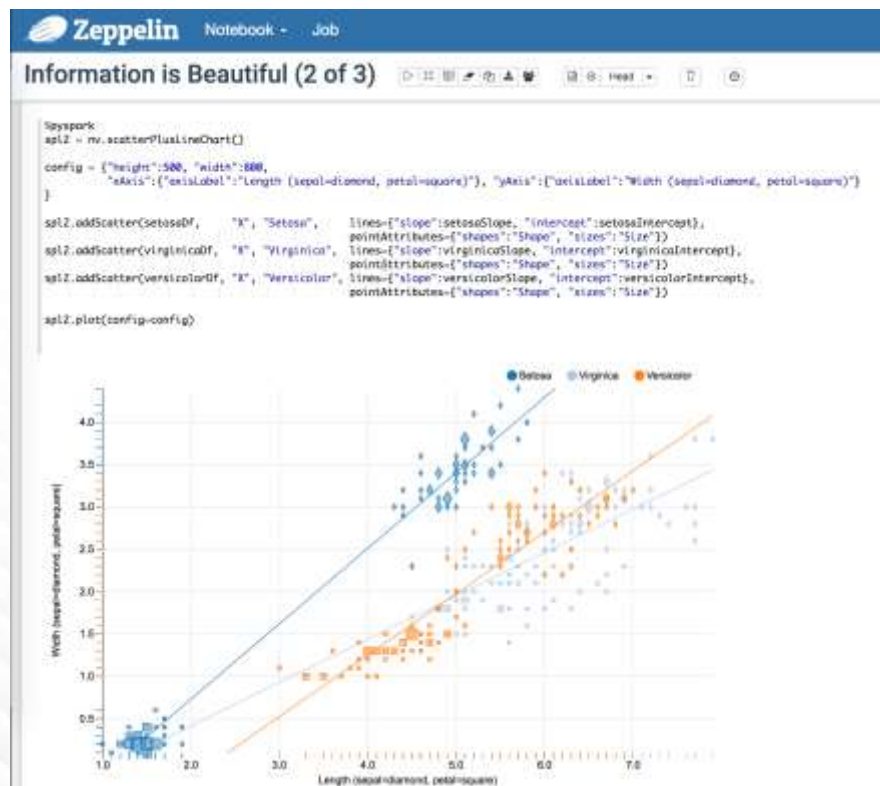
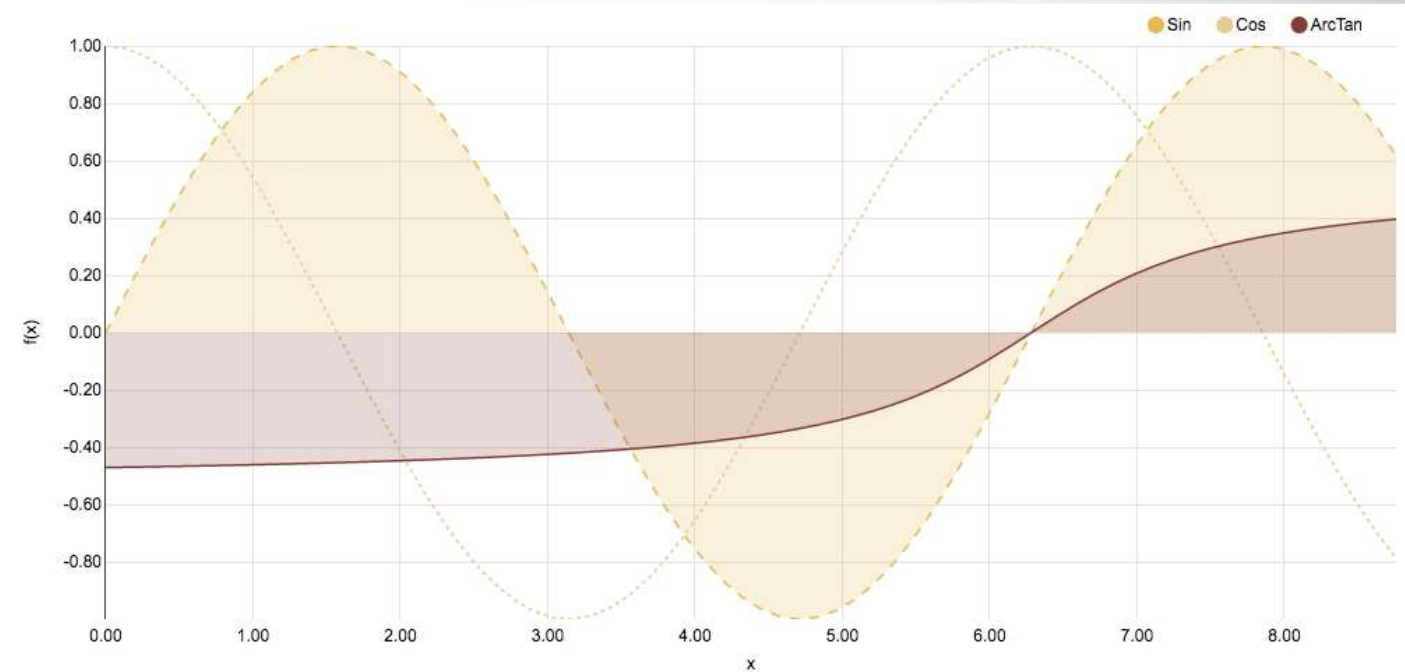
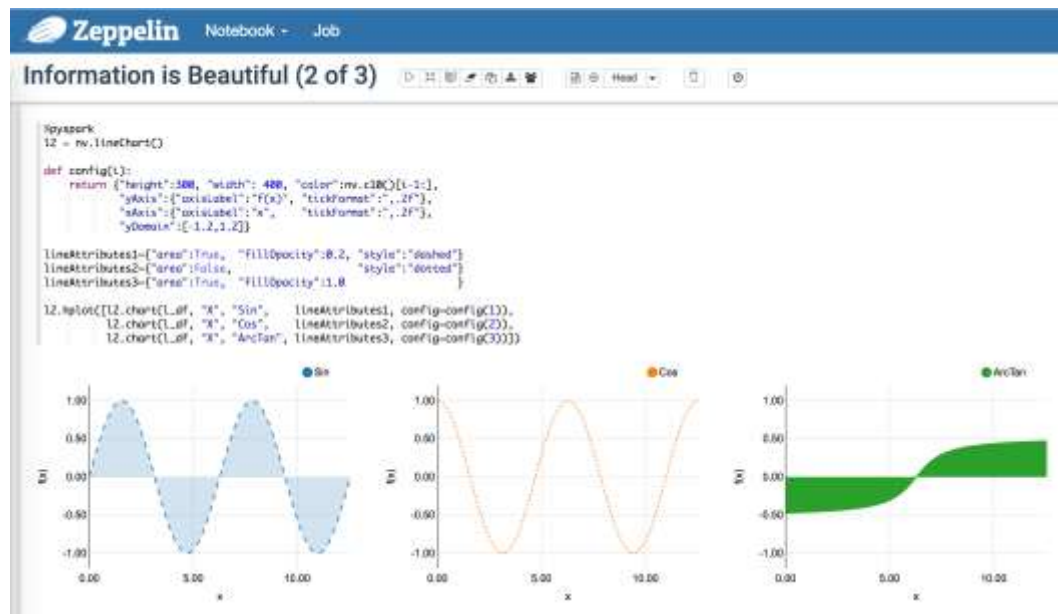
Features

- Ad-hoc experimentation
- Deeply integrated with Spark + Hadoop
- Supports multiple language backends
- Incubating at Apache

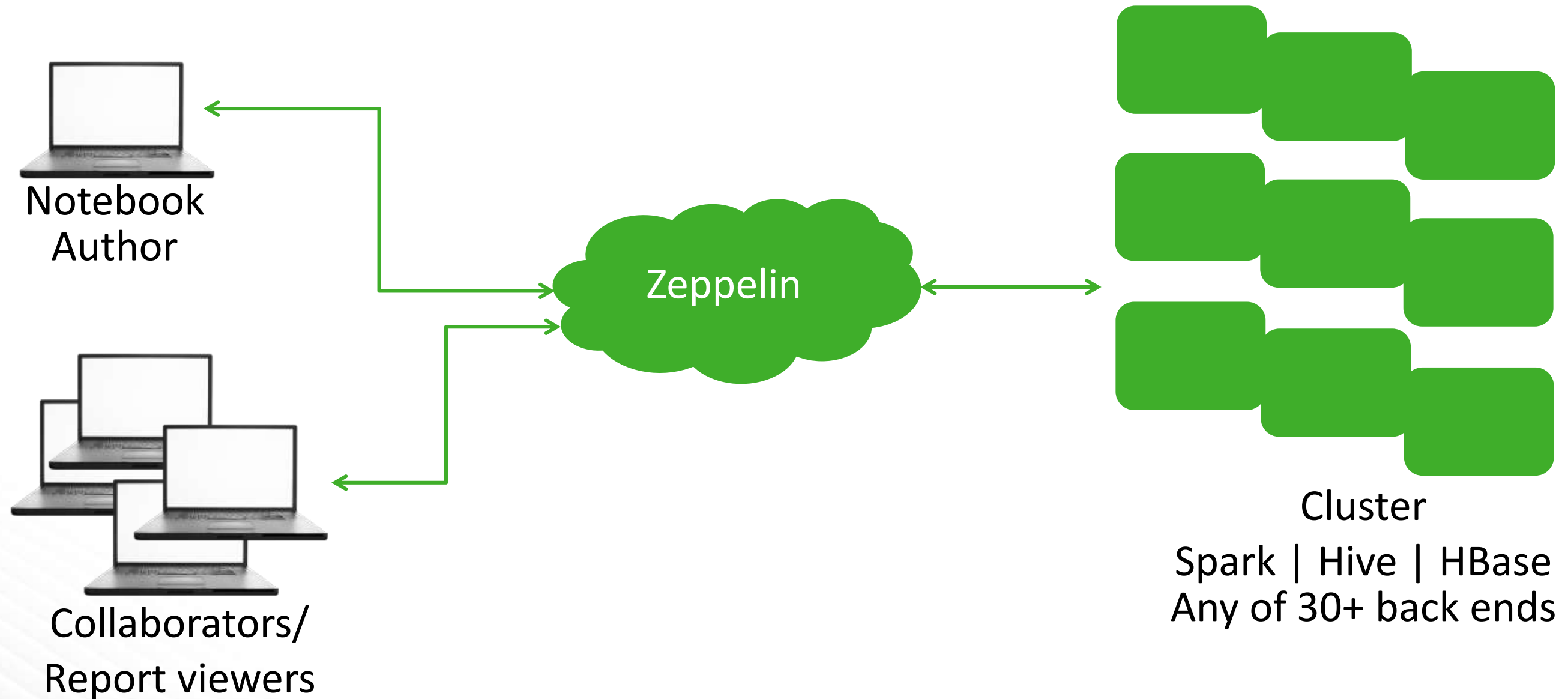
Use Case

- Data exploration and discovery
- Visualization
- Interactive snippet-at-a-time experience
- “Modern Data Science Studio”

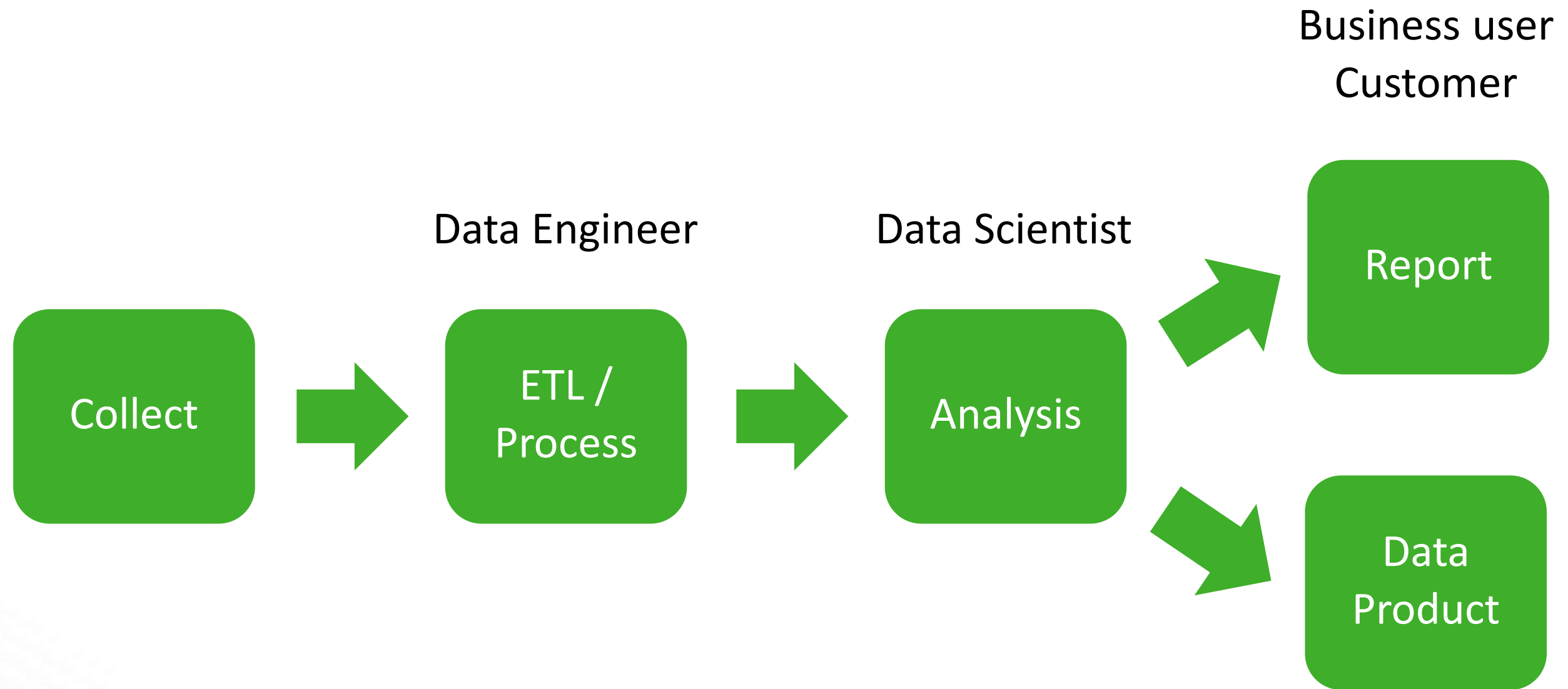




How does Zeppelin work?



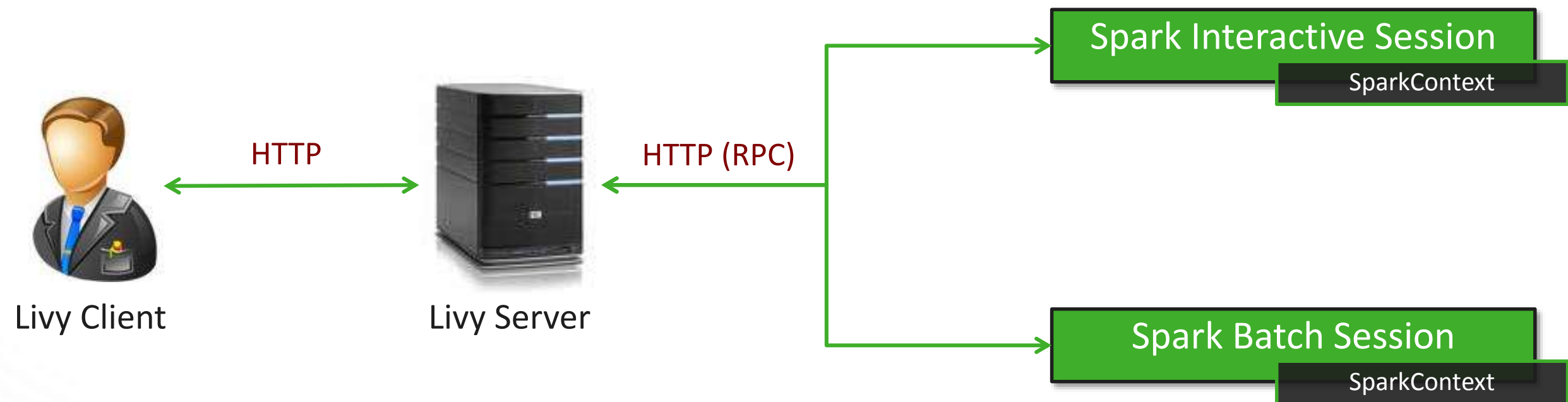
Big Data Lifecycle



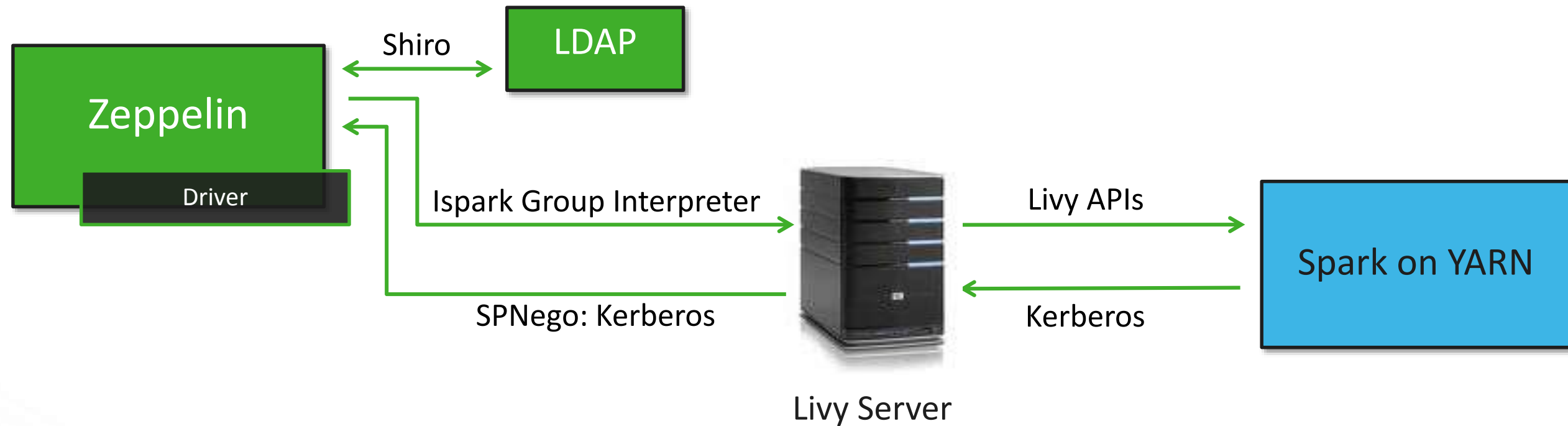
Zeppelin Multitenancy

Livy

- ◆ Livy is the open source REST interface for interacting with Apache Spark from anywhere
- ◆ Installed as Spark Ambari Service



Security Across Zeppelin-Livy-Spark



Reasons to Integrate with Livy

◆ Bring Sessions to Apache Zeppelin

- Isolation
- Session sharing

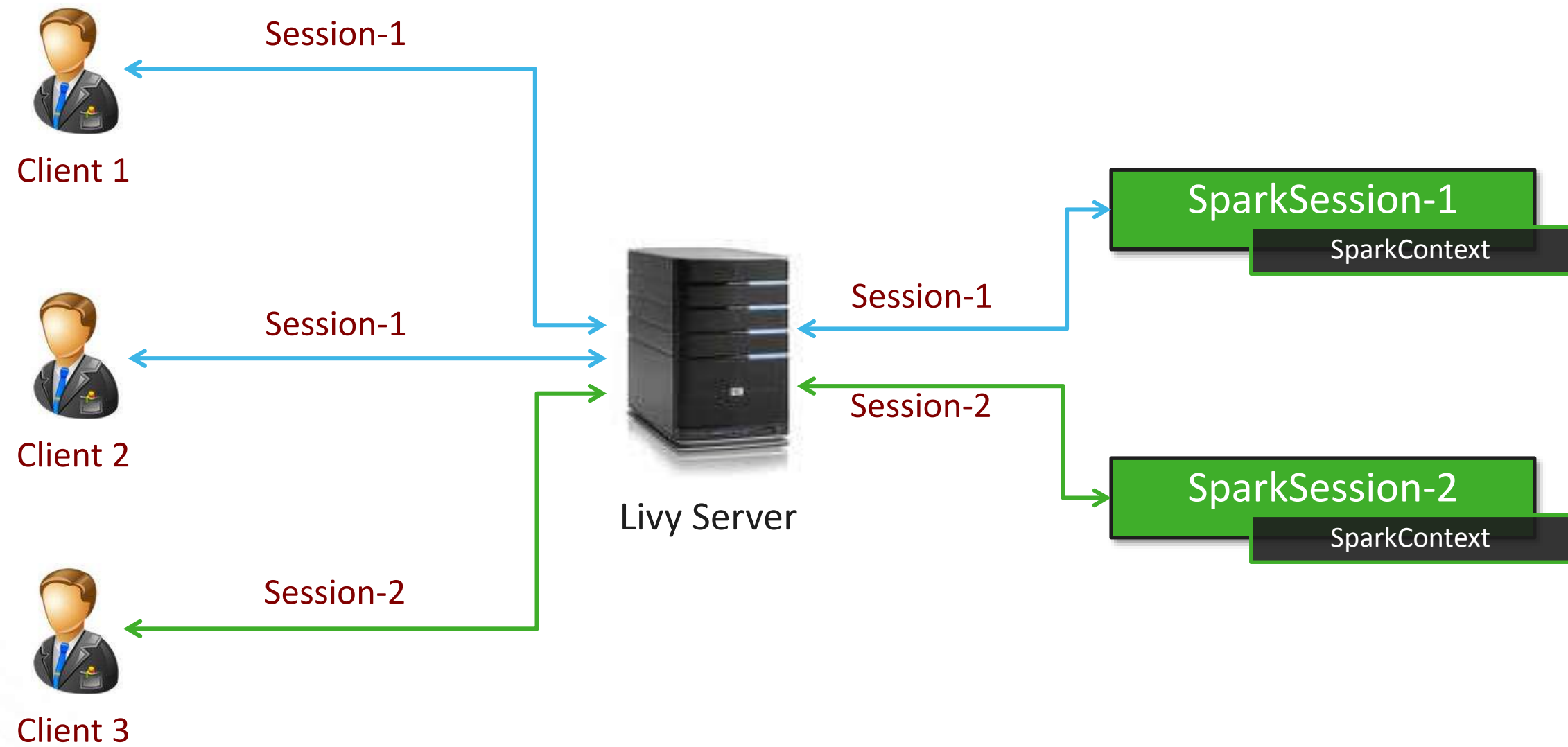
◆ Enable efficient cluster resource utilization

- Default Spark interpreter keeps YARN/Spark job running forever
- Livy interpreter recycled after 60 minutes of inactivity (controlled by `livy.server.session.timeout`)

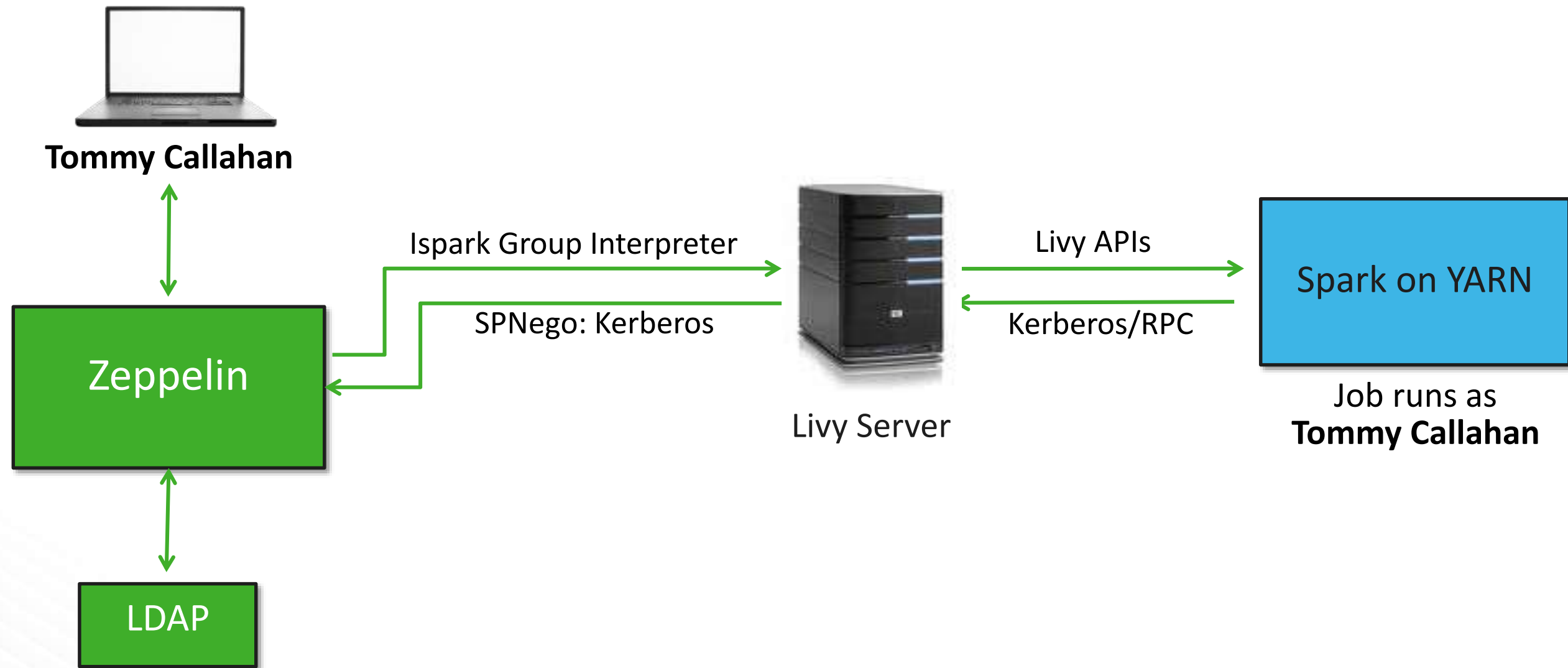
◆ To Identity Propagation

- Send user identity from Zeppelin > Livy > Spark on YARN

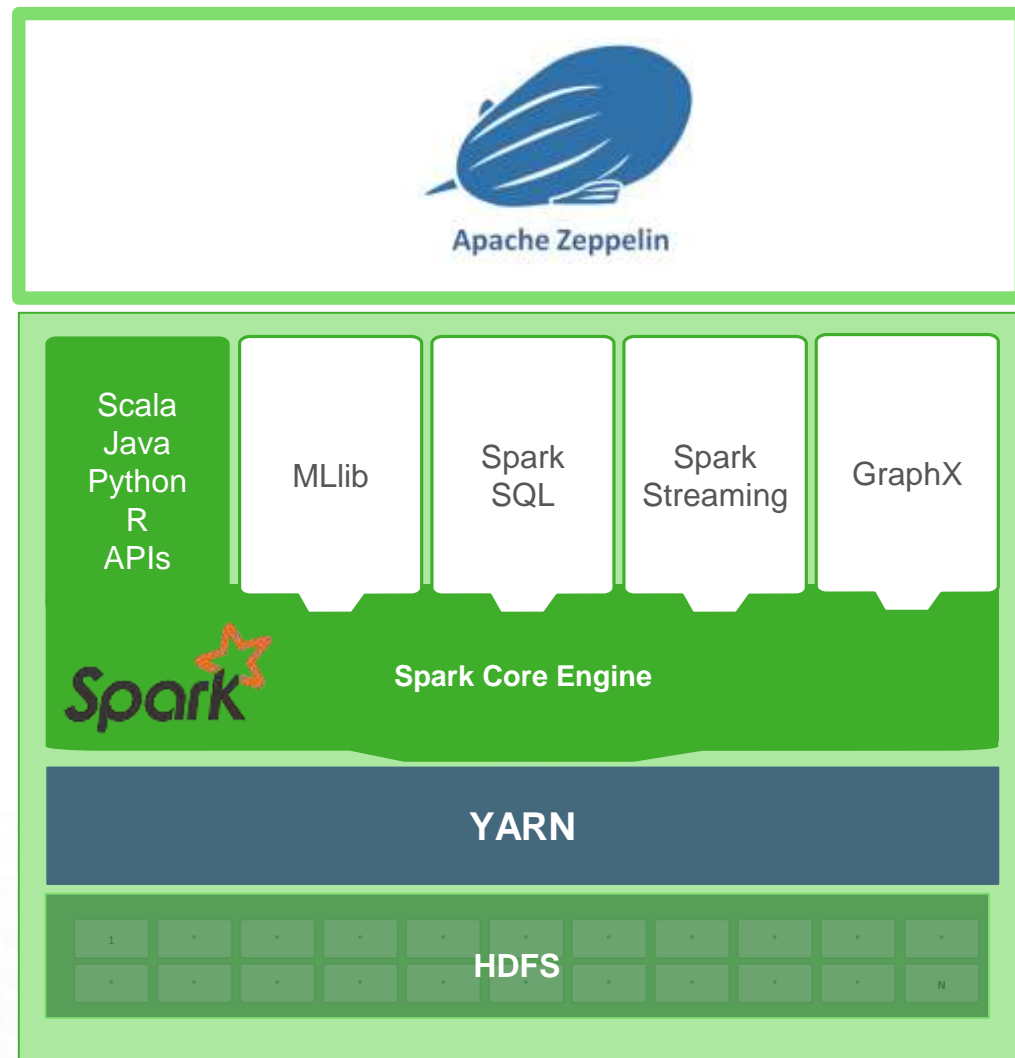
SparkSession Sharing



Apache Zeppelin + Livy End-to-End Security

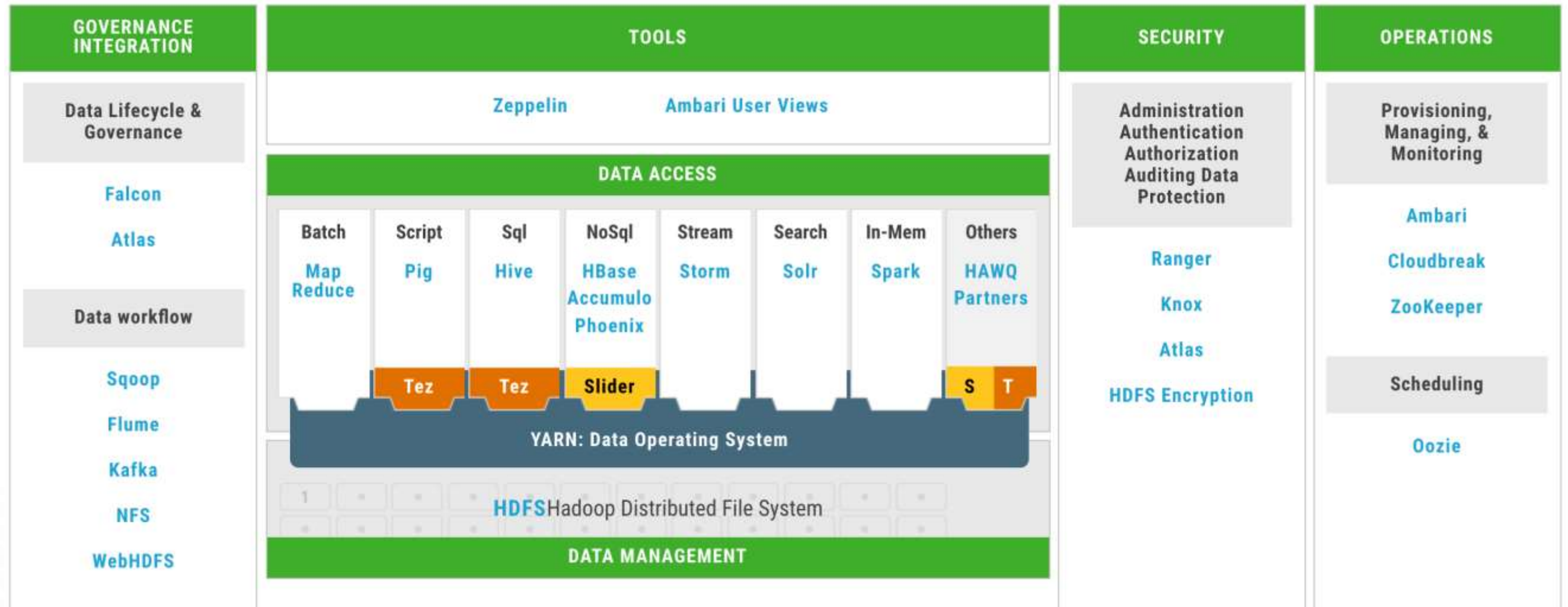


HDP Basics

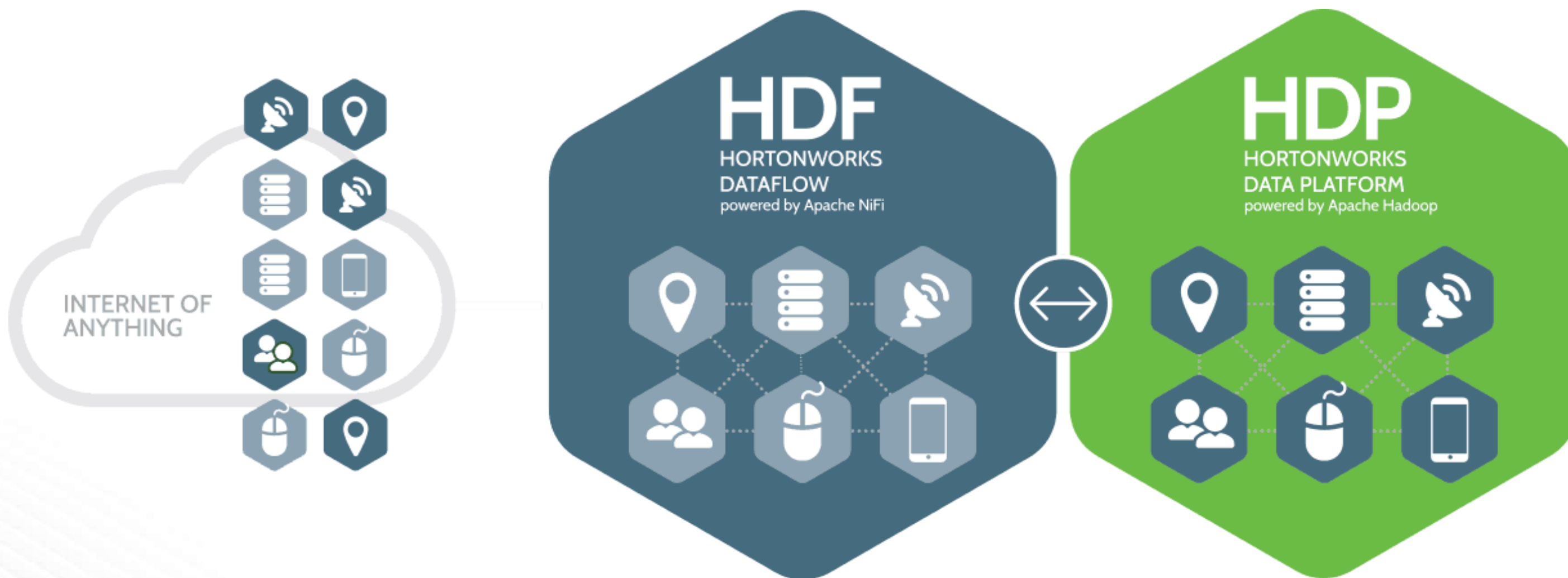


- ◆ Zeppelin → Interactive notebook
- ◆ Spark
- ◆ YARN → Resource Management
- ◆ HDFS → Distributed Storage Layer (4M files)
 - Future: Ozone object store

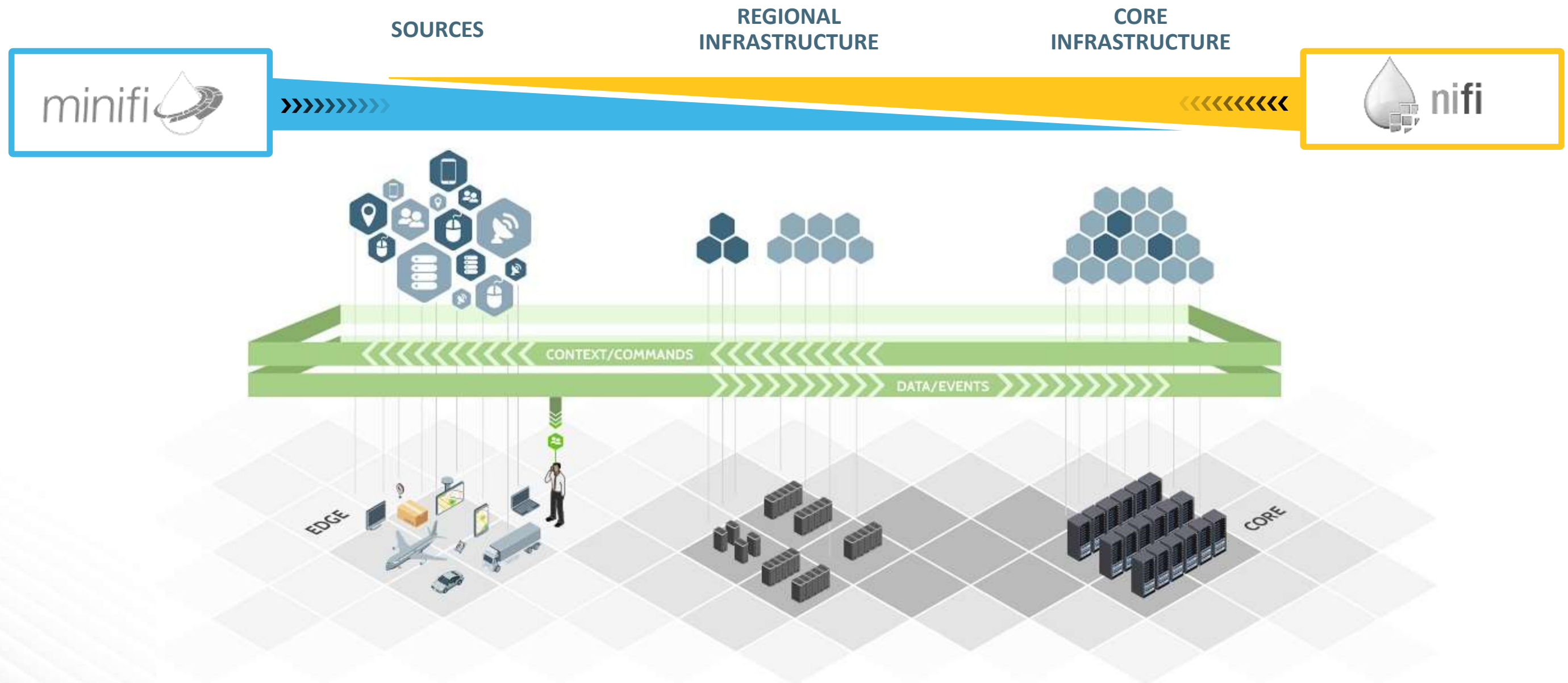
Hortonworks Data Platform



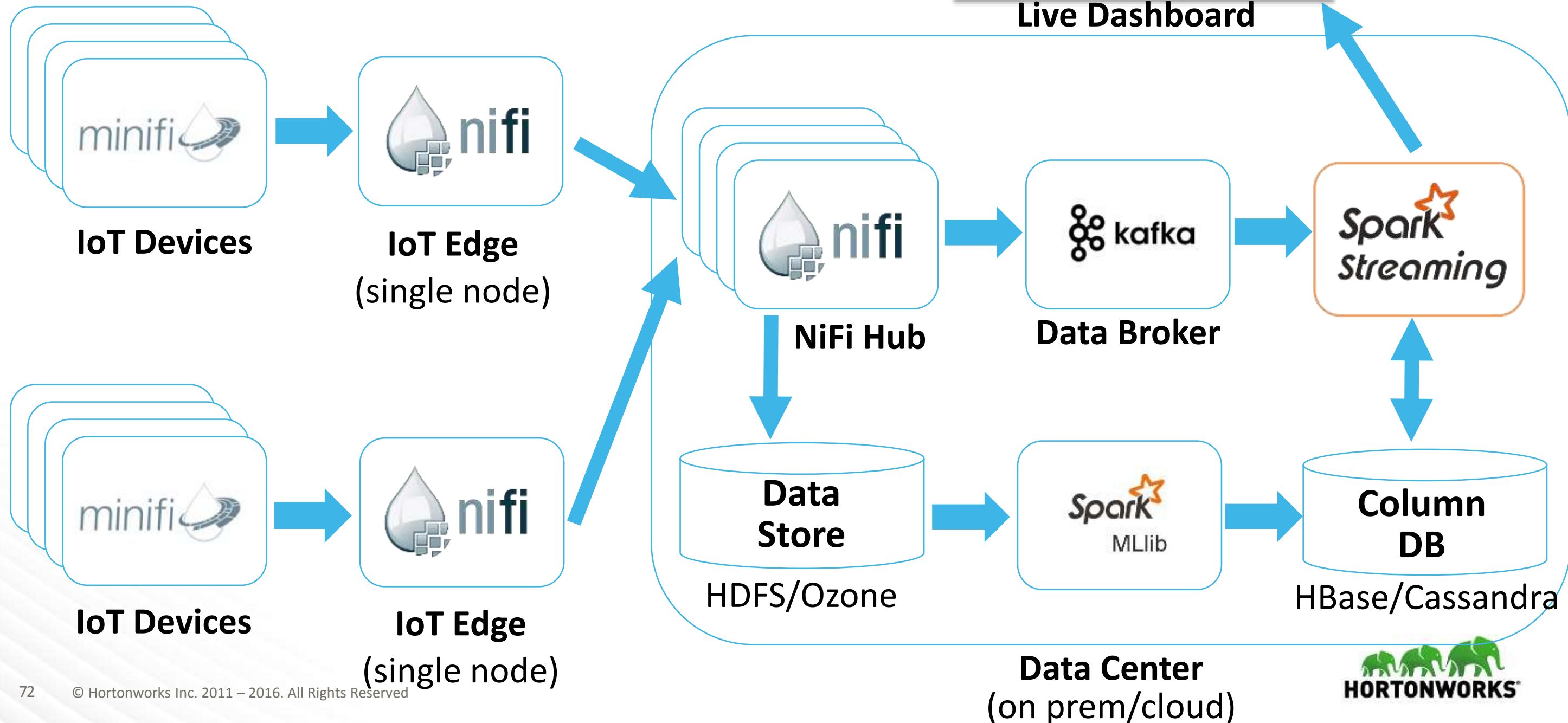
Sample Architecture



Managed Dataflow



High-Level Overview



Spark 2.x & HDP 2.x

What's New?

What's New

◆ Future HDP / Spark 2.3

- Spark **Structured Streaming latency** in single-digit **milliseconds** in **continuous mode** in stream processing (instead of 100ms we'd normally see with micro batching)
- stream-to-stream joins
- PySpark boost by improving performance with pandas UDFs
- runs on Kubernetes clusters by providing native support for Apache Spark applications

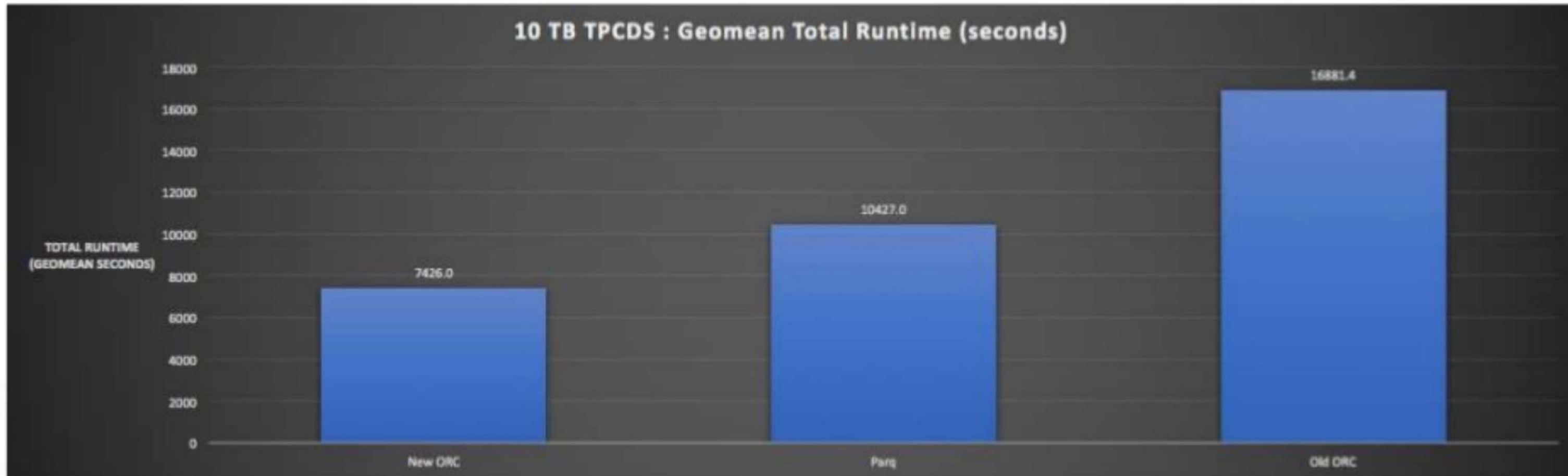
◆ HDP 2.6.4 / Spark 2.2

- Structured Streaming GA
- Yahoo! Benchmark: 65M rec/s
- ORC feature & performance improvements → Parquet Parity

◆ HDP 2.6.3 / Spark 2.1

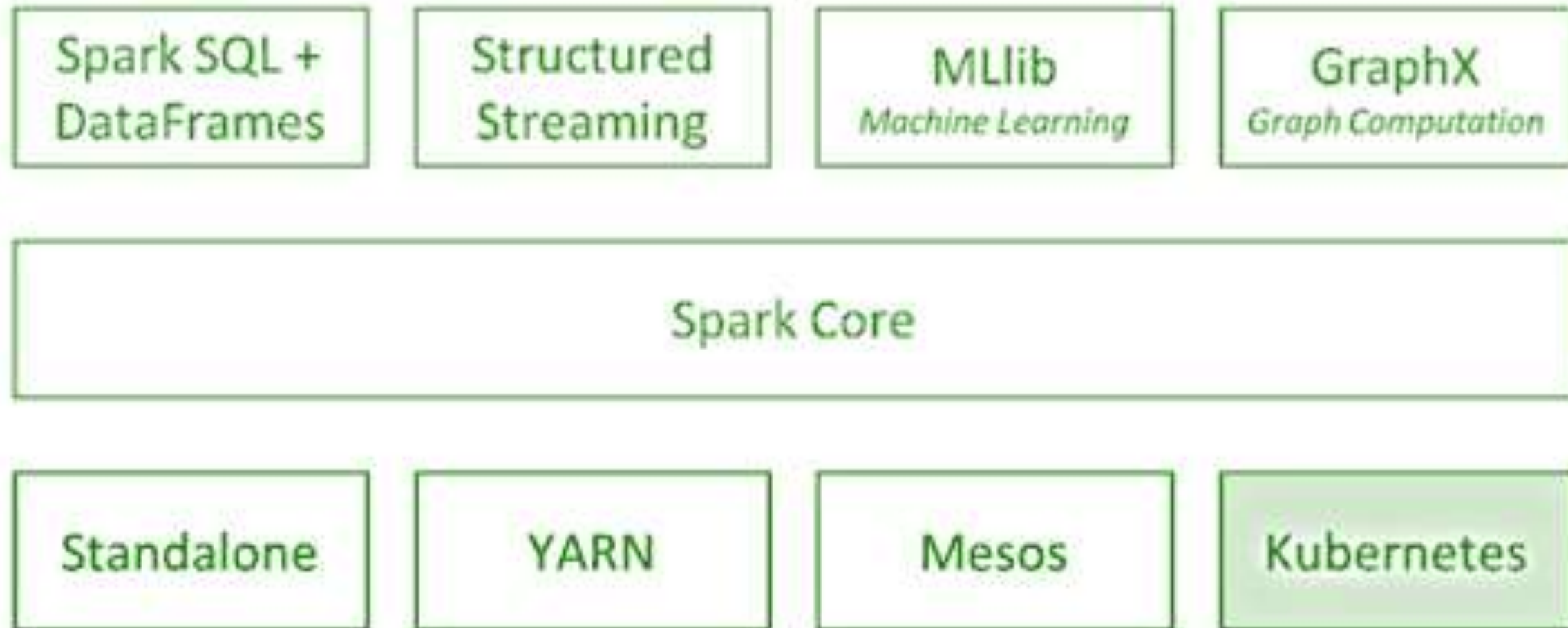
- Spark SQL Ranger integration for row and column security
- DataSet API GA
- GraphX GA

Comparison of New ORC vs Old ORC vs Parquet (10 TB Scale)



- * Total runtime of 74 queries in TPC-DS
- * 10 TB Scale (15 executors, 170 GB, 25 cores)

Spark 2.3



Spark SQL

Structured Data

Spark Streaming

Near Real-time

Spark MLlib

Machine Learning

GraphX

Graph Analysis



DSX + HDP



Data Science Experience (DSX) Local
Enterprise Data Science platform for teams

REST interface

Hortonworks Data Platform (HDP)
Enterprise compute (Spark/Hive) & storage
(HDFS/Ozone)

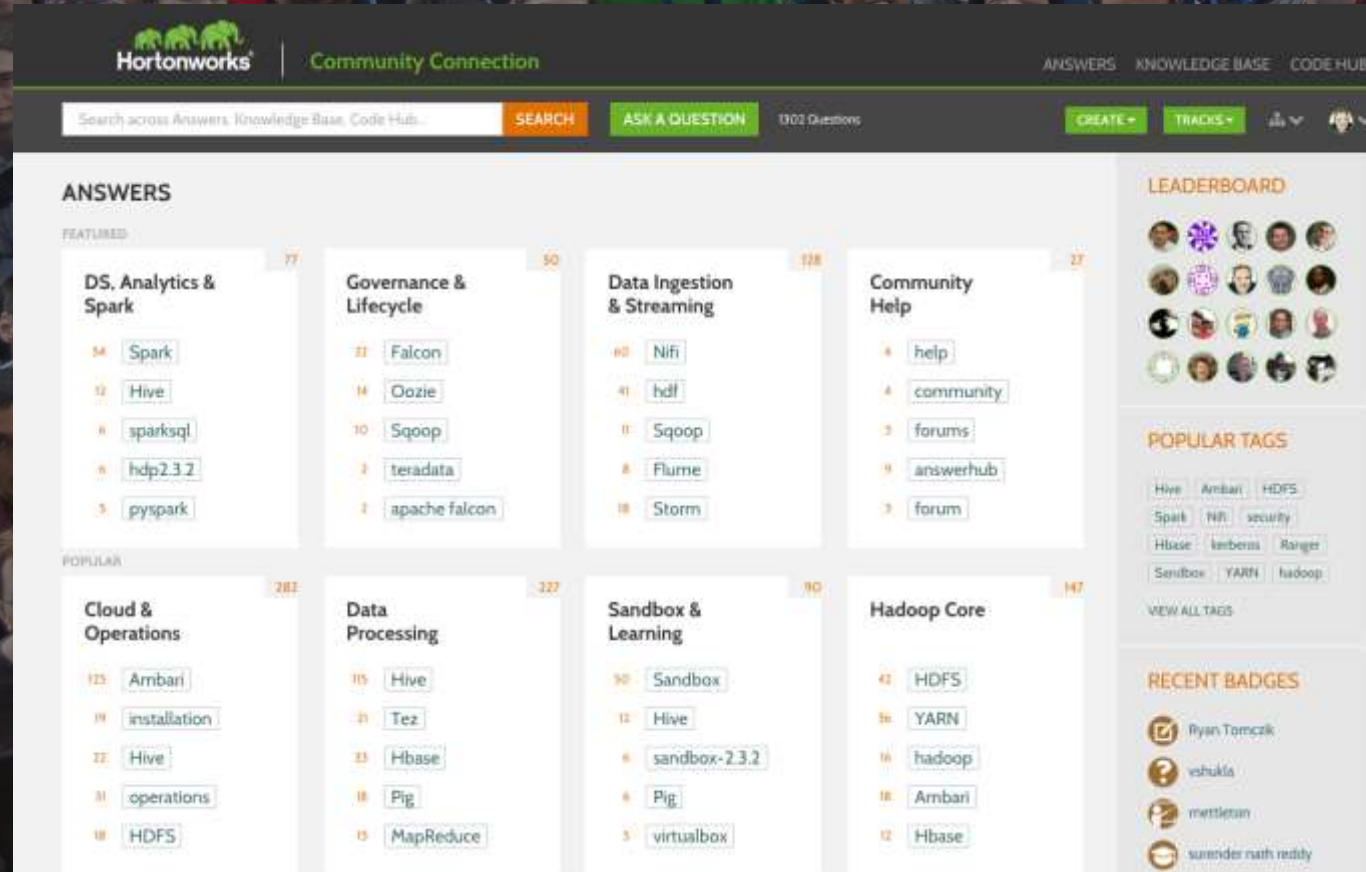


Lab



Hortonworks Community Connection

community.hortonworks.com



- Full Q&A Platform (like StackOverflow)
- Knowledge Base Articles
- Code Samples and Repositories



Community Engagement

community.hortonworks.com

20k+

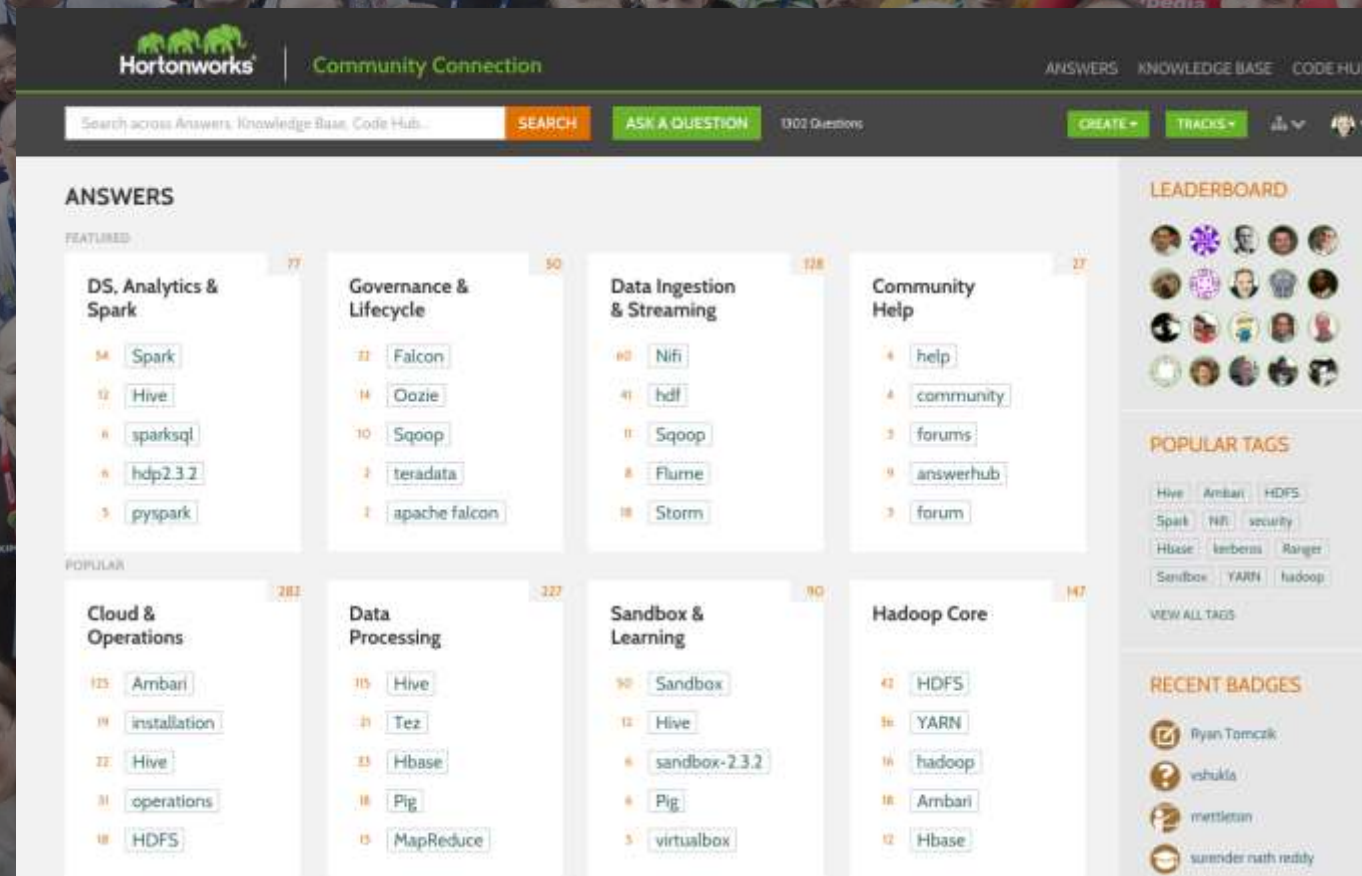
Registered Users

45k+

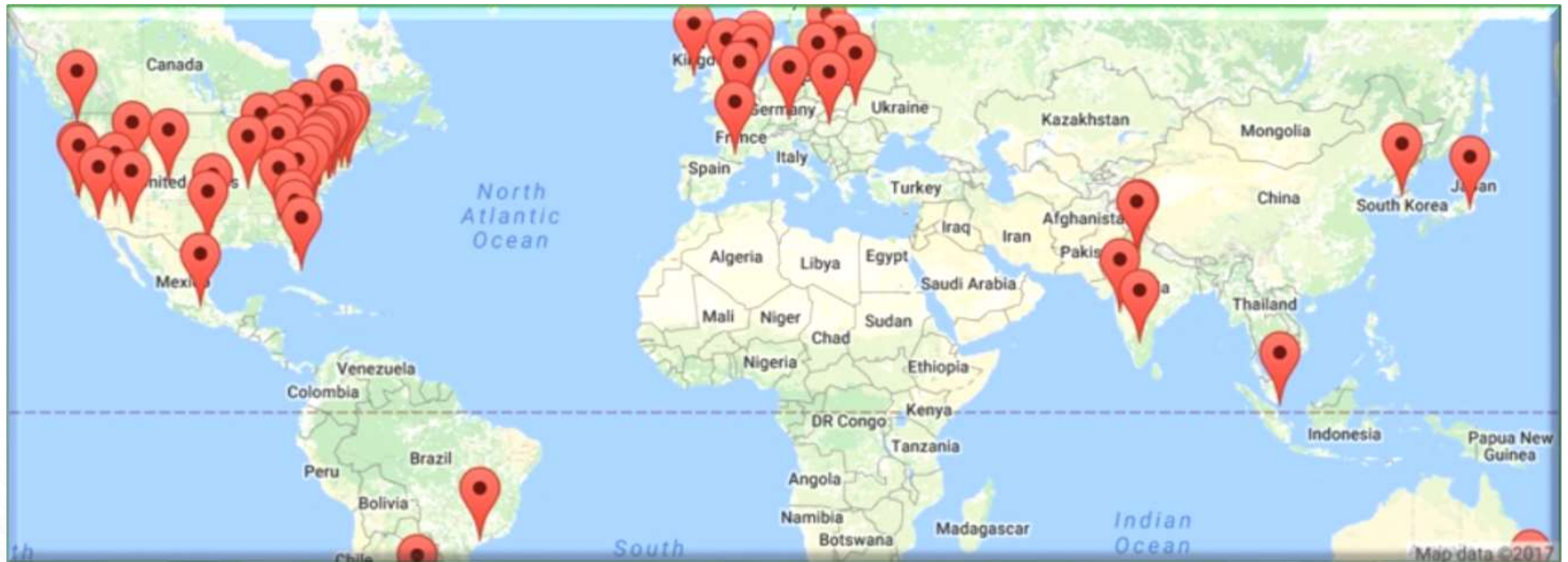
Answers

100k+

Technical Assets



Future of Data Meetups



Thanks!

Robert Hryniewicz
@RobHryniewicz

