insight into value

BIG DATA ANALYTICS
BUSINESS INTELLIGENCE
INFORMATION MANAGEMENT
PERFORMANCE MANAGEMENT

Conteúdo Programático

- BIG DATA
- CLOUD COMPUTING
- APACHE HADOOP
- APACHE SPARK
- DATABRICKS

DATA ENGINEERING

Keyrus • Ago. 25, 2020

The Presenter

ALAN SILVA

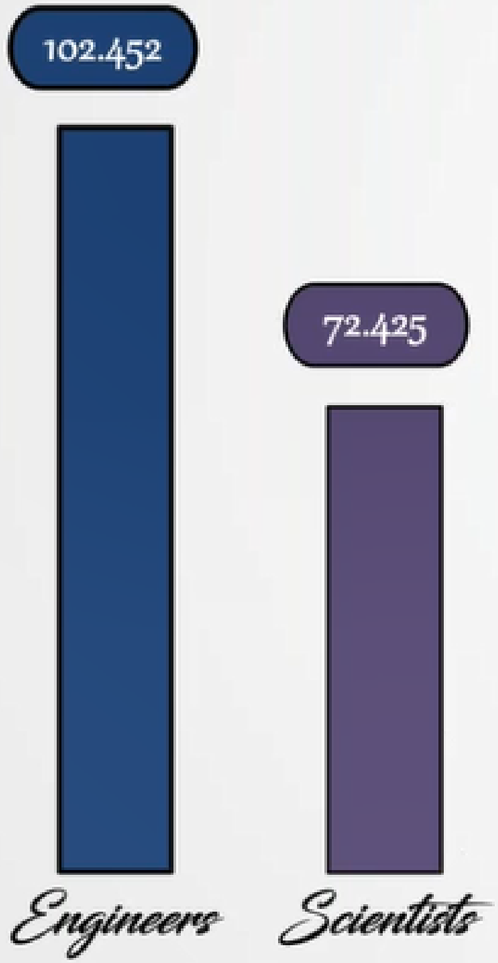
Engenheiro de Dados

Pós Graduado em Ciência de Dados e Big Data
Engenheiro da Computação
Linkedin: www.linkedin.com/in/alancarlossilva

ORACLE Certified Associate

ORACLE Certified Specialist

Vagas Engineers vs. Scientists



Companies

Salaries

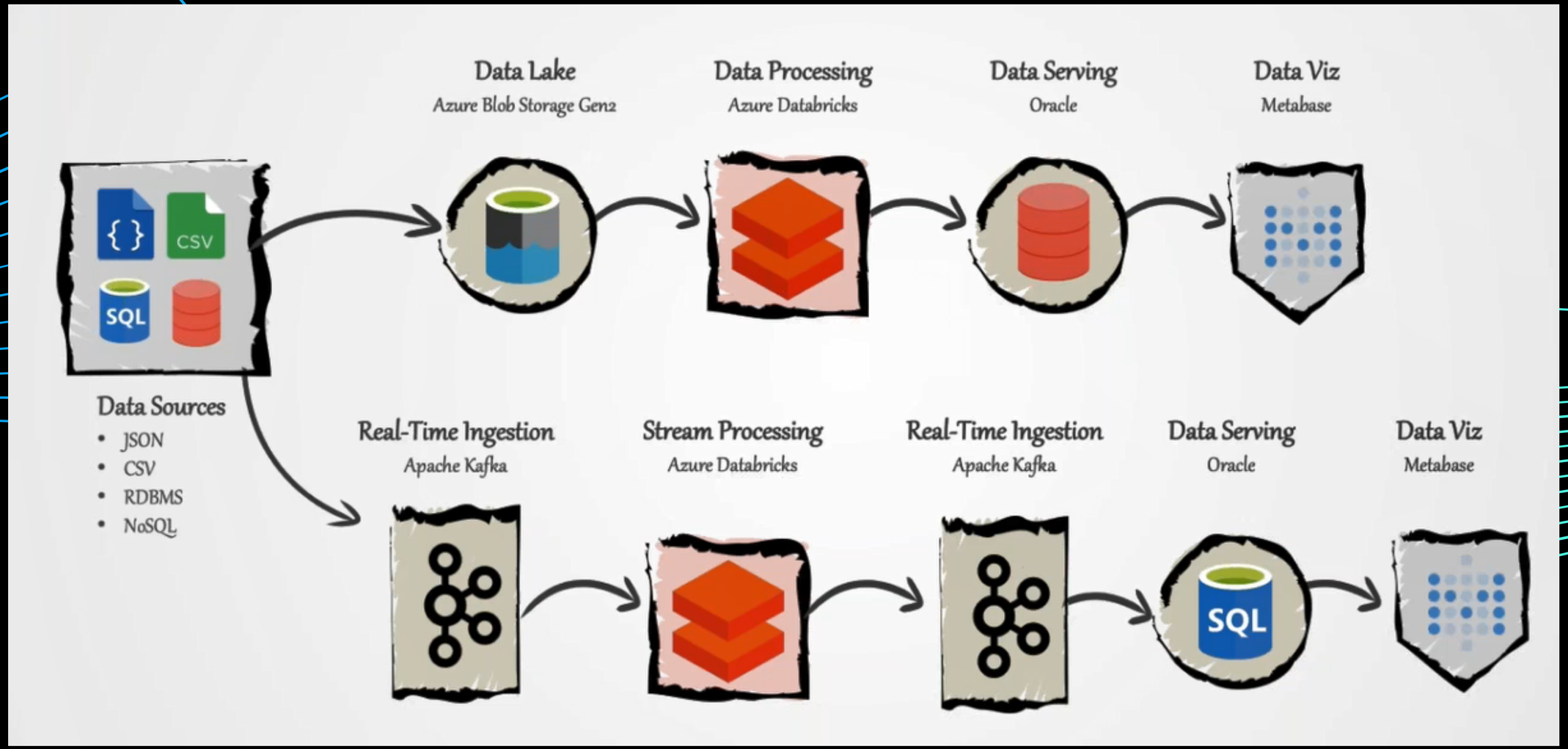
- Data Engineer = $110,000 ~ $155,000
- Data Scientist = $130,000 ~ $185,000

Data Engineer Resume

- Knowledge = Data Warehouse, ETL/ELT, Data Modeling
- Cloud Computing = AWS, GCP, Azure
- Programming Language = Python, Scala, BASH
- Big Data = Apache Hive, Apache Spark, Apache Kafka, Apache Airflow, Apache Druid
- RDBMS = MySQL, PostgreSQL
- NoSQL = Cassandra, MongoDB, ElasticSearch
- OS = Linux

Lambda Architecture

Batch-Layer



Fast-Layer

Big Data

Netflix, Inc.

137 million users worldwide with consumption of 25% of the world's internet bandwidth

Spotify

191 million users worldwide with more than 30 million of songs available

Airbnb, Inc.

income of $107 millions with average of 2m+ of people staying in places

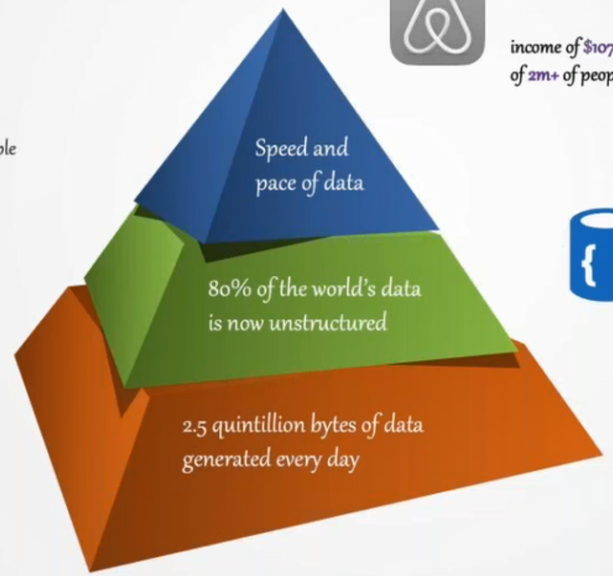
Lyft, Inc.

1m+ million of rides per day and 30M+ of users worldwide

Apache Kafka [2014]

speed of data generation & processing large datasets, ability to ingest data as fast as possible

Batch, Near, Real-Time



NoSQL [2009]

type & nature of data, different data sources & mappings, easier for developers

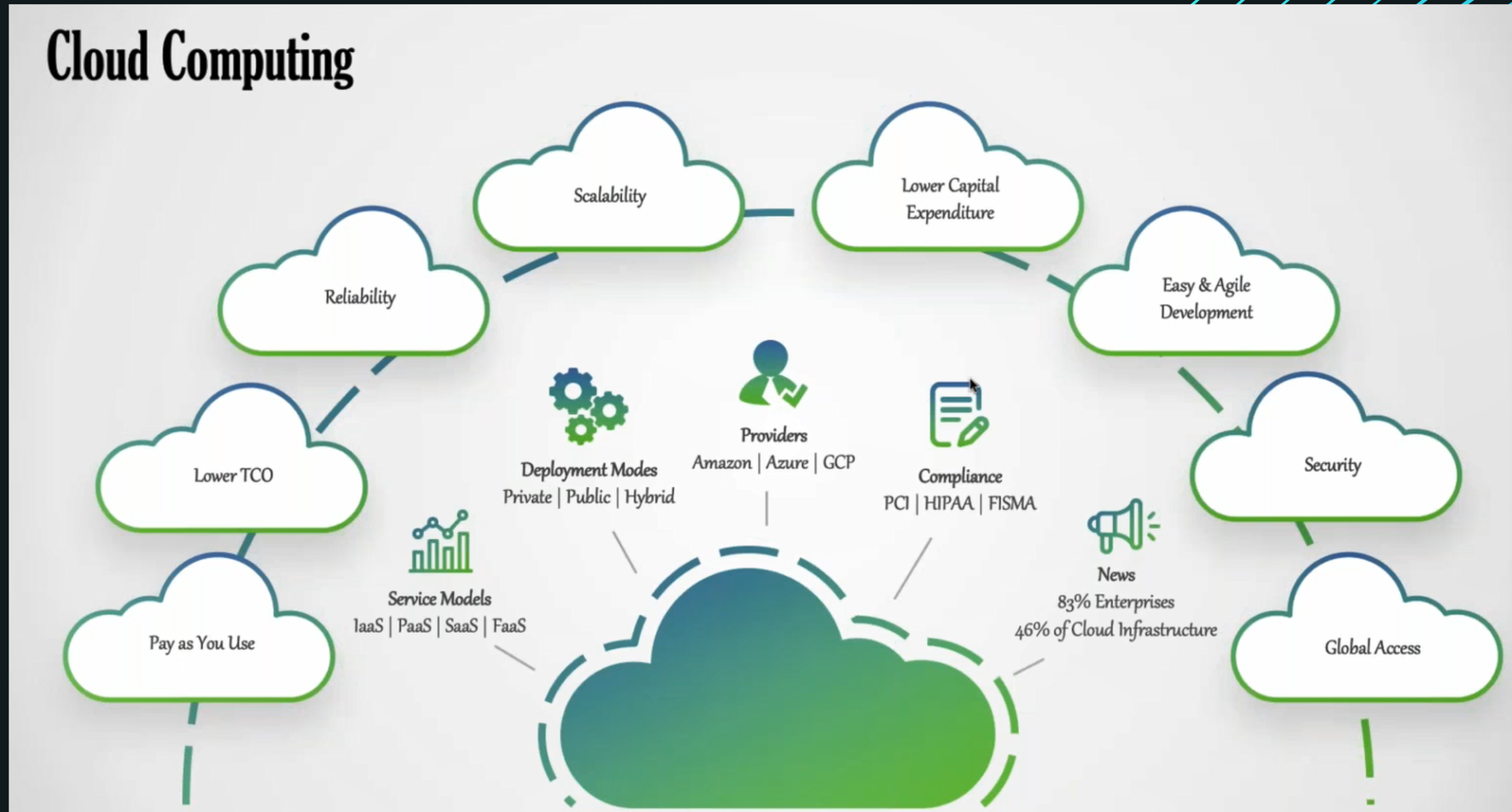
Key-Value Pairs, Column-Family, JSON, Graph

Apache Hadoop [2006]

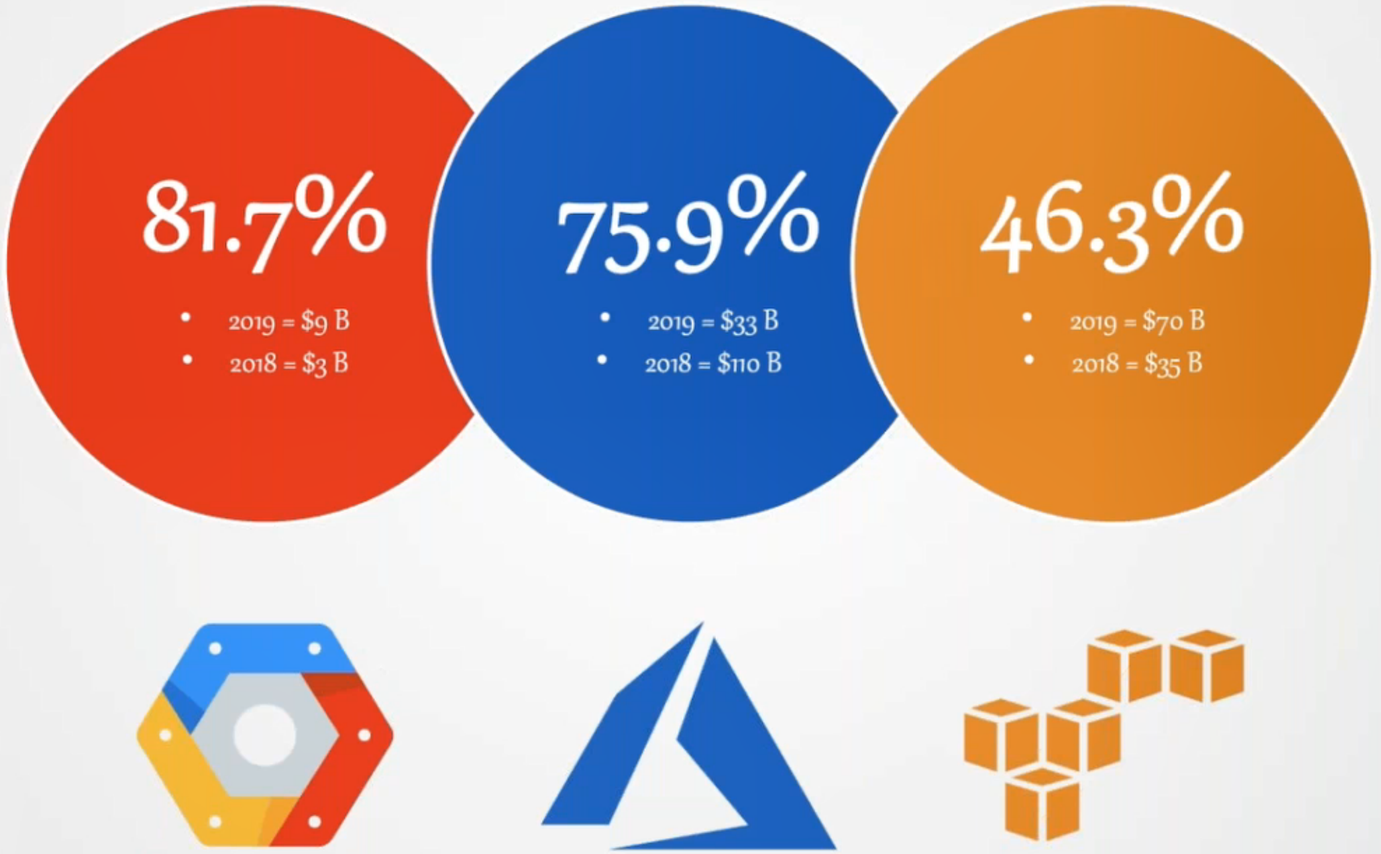
quantity of generated & stored data, size of data & value of potential insights

MB | GB | TB | PB

Cloud Computing



Google Cloud Platform vs. Microsoft Azure vs. Amazon Web Services

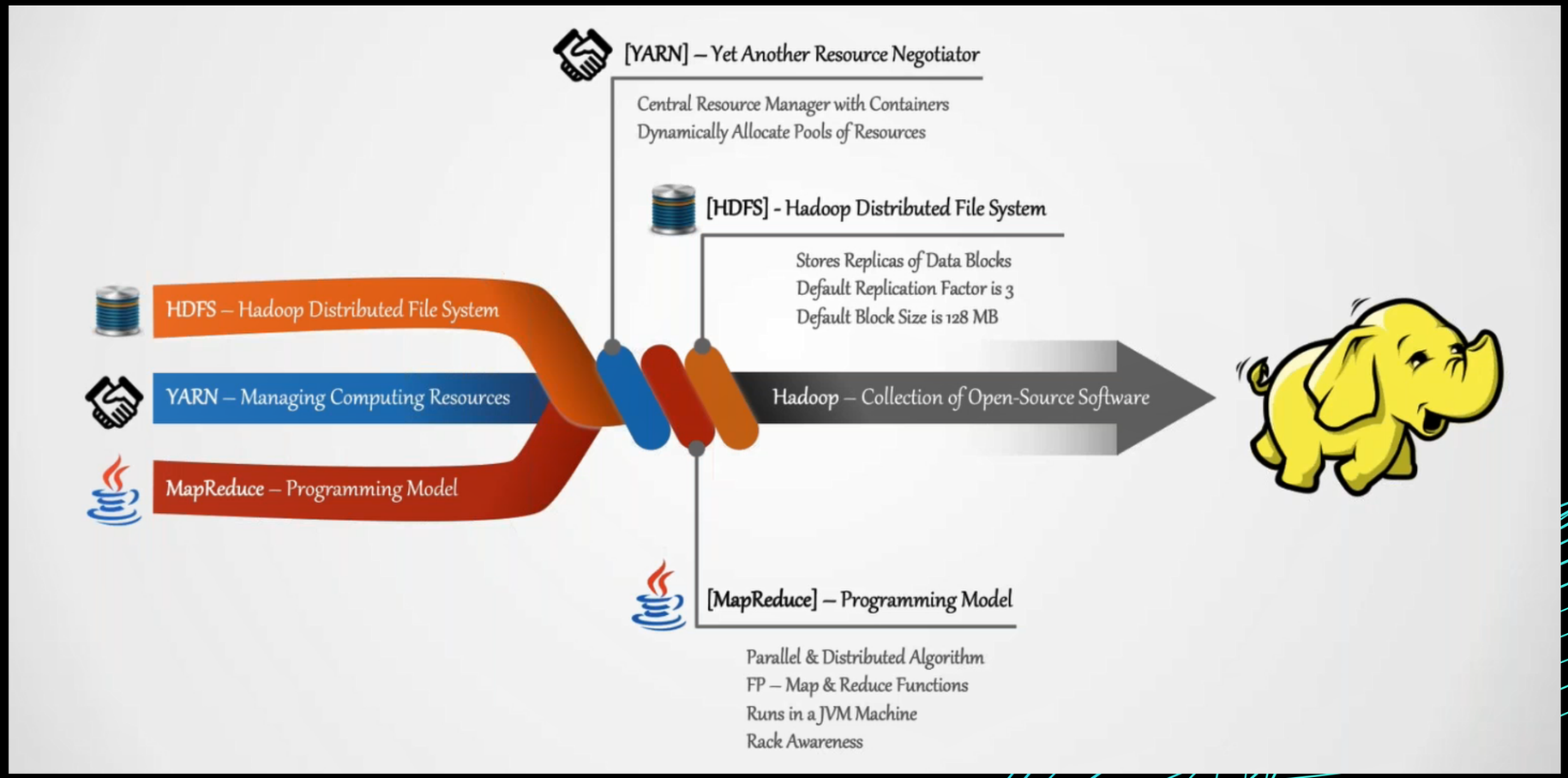


#Spark

What is the Apache Spark?

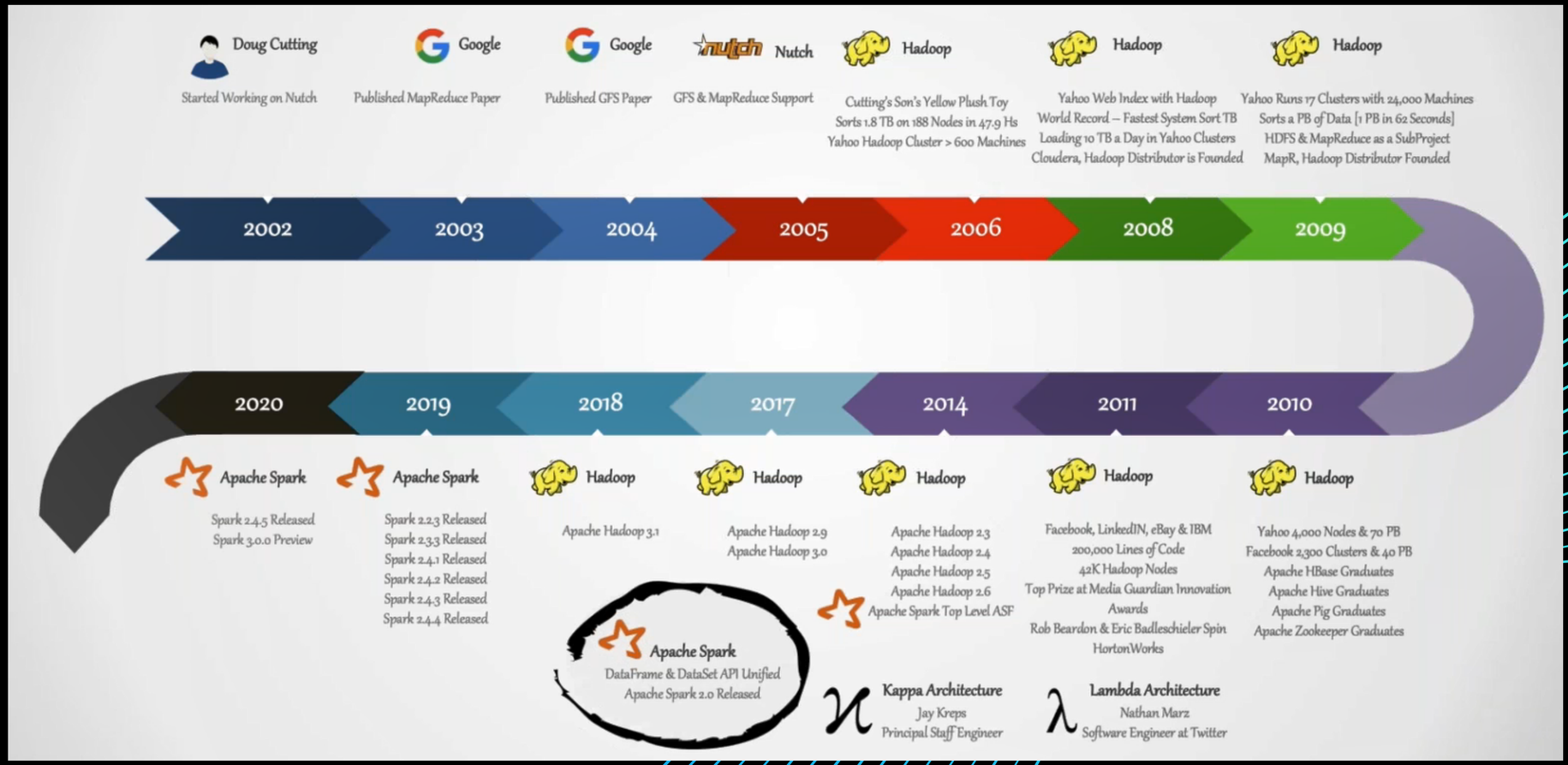
Precisamos entender o Hadoop para entendermos o Spark

Apache Hadoop [Fundamentals]

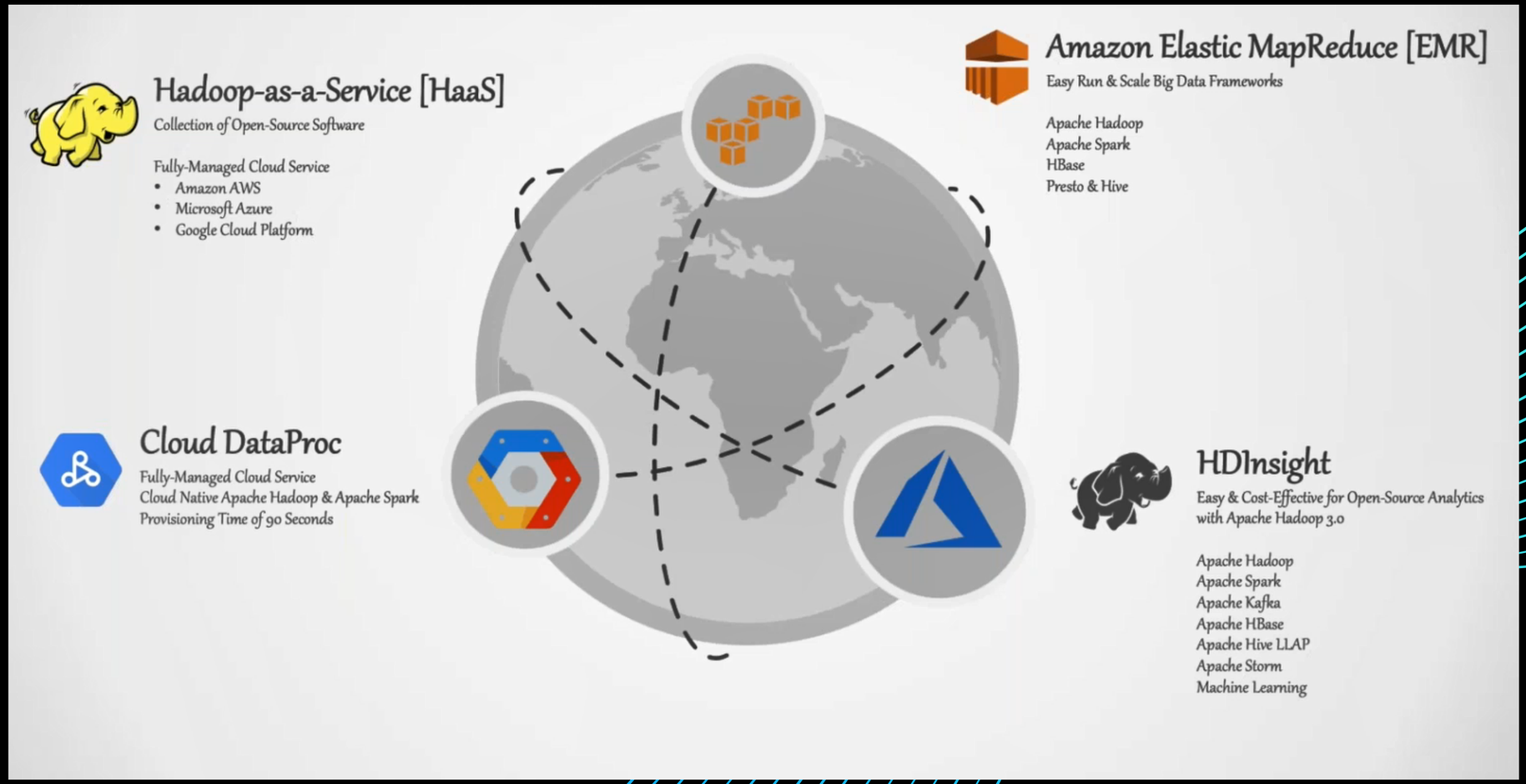


Ecosystem of Hadoop Animal Zoo [2006~2014]

History of Apache Hadoop & Apache Spark



Big Data-as-a-Services [BDaaS]



Develop Java MapReduce Program for Apache Hadoop on [HDInsight]

Apache Spark [Fundamentals]

Apache Spark

Open-Source Distributed Cluster-Computing Framework
Implicit Data Parallelism & Fault Tolerance
Optimized for Memory Computation
Written In - Scala
100x ~ MapReduce Jobs & 10x – Disk-Based Operations

History

University of California, Berkeley's AMPLab
Open-Sourced in 2014 – Top-Level Apache Project
Databricks – New World Record in Large Scale Sorting [2014]
Ali Ghodsi | Reynold Xin | Matei Zaharia ~ **Databricks**
1,000 Contributors in 2015

Key Capabilities

Unified Stack for Interactive, Streaming & Predictive Analysis
Batch & Streaming in an Unified Platform
Designed for Large-Scale Data Processing

Performance

Daytona Gray
- 100 TB in **23 Minutes** with 206 EC2 VMs
- 100 TB in 72 Minutes with 2.100 Nodes

Use-Cases

AirBnB – ML & Streaming on Mobile App

Uber – Continuous ETL Pipeline by Processing Apache Kafka Data and Storing on HDFS

Netflix – Data Analytics Computation & ML at Scale

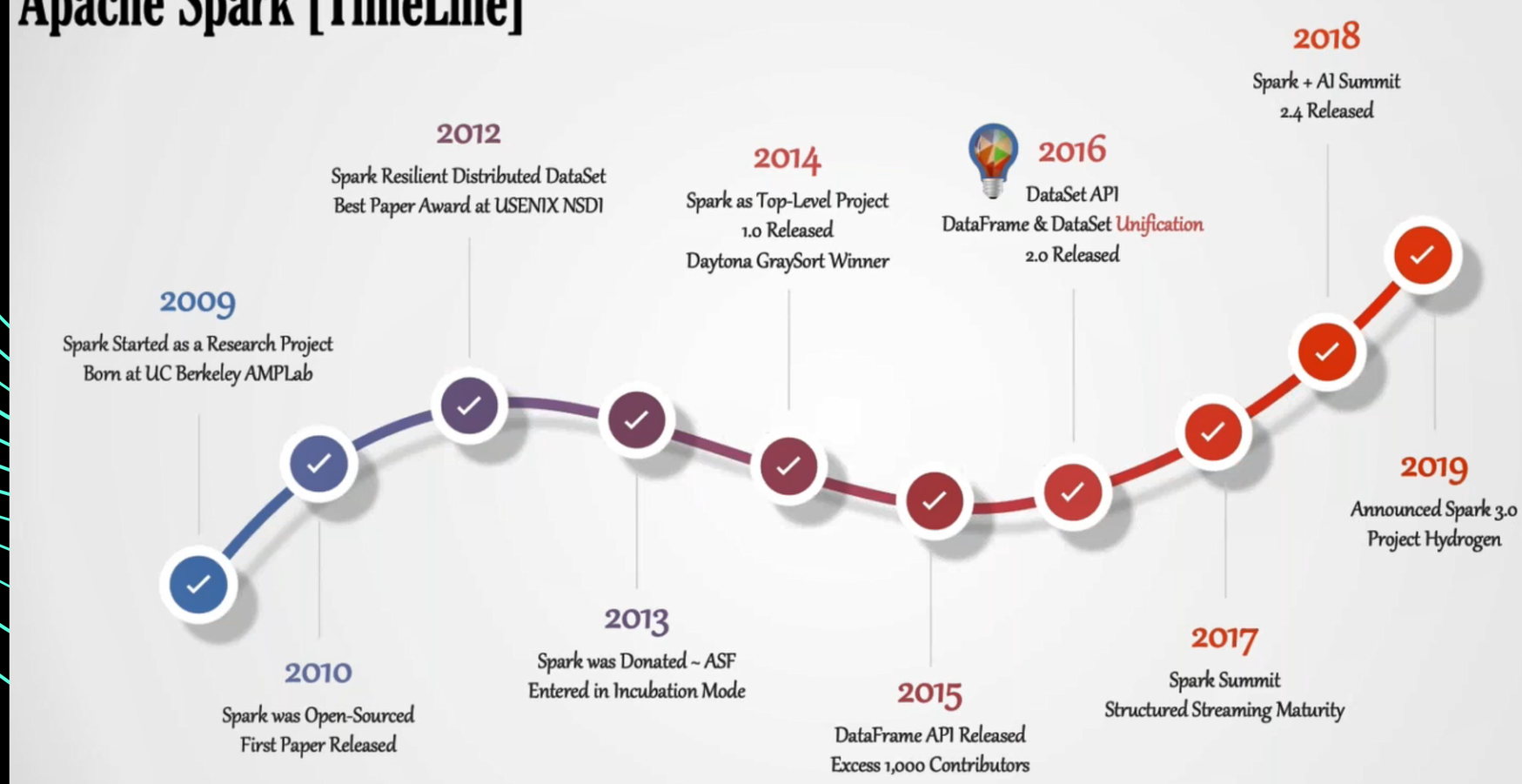
Core APIs

SQL
Java
Scala
Python
R

Processing Structures

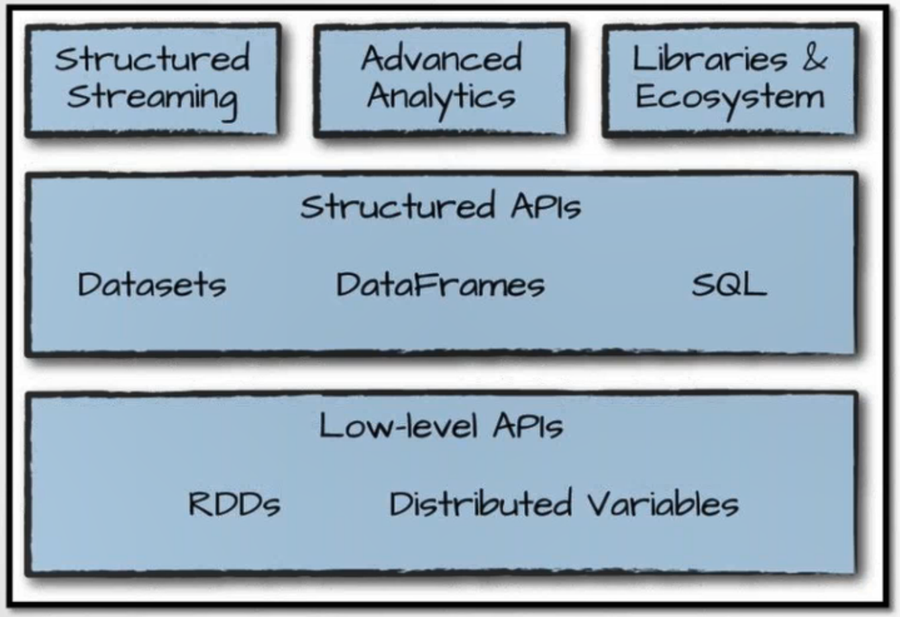
RDD – Resilient Distributed DataSet
Spark Streaming – Processing Data Streams using DStreams
Spark-SQL, DataSets & DataFrames – Processing Structured Data
Structured Streaming – Processing Structured Data Streams

Apache Spark [TimeLine]



Apache Spark [Components]

developers stopped making individual processors faster and switched toward adding more parallel cpu cores all running at the same speed. this change meant that suddenly applications needed to be modified to add parallelism in order to run faster, which set the stage for new programming models such as apache spark.



High-Level APIs

the fundamental abstraction that you will use to write most of your data flows.

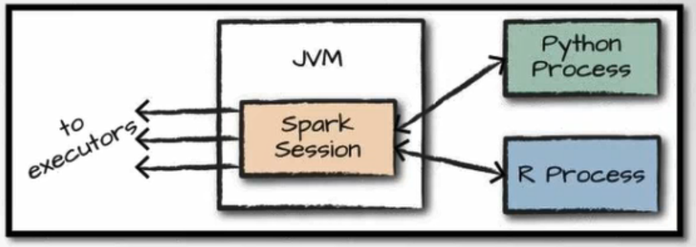
- fundamentally, spark is a distributed programming model in which the user specifies transformations, these transforms build a directed acyclic graph of instructions
- actions, begins the process of executing theses dags as a single job by breaking down into stages and tasks to execute across the cluster

Low-Level APIs

virtually everything in spark is built on top of RDDs.

- you need some functionality that you cannot find in higher level apis for example a very tight control over physical data placement across the cluster
- you need to maintain some legacy codebase written in RDDs
- add some custom shared variable manipulation

Apache Spark [Architecture]

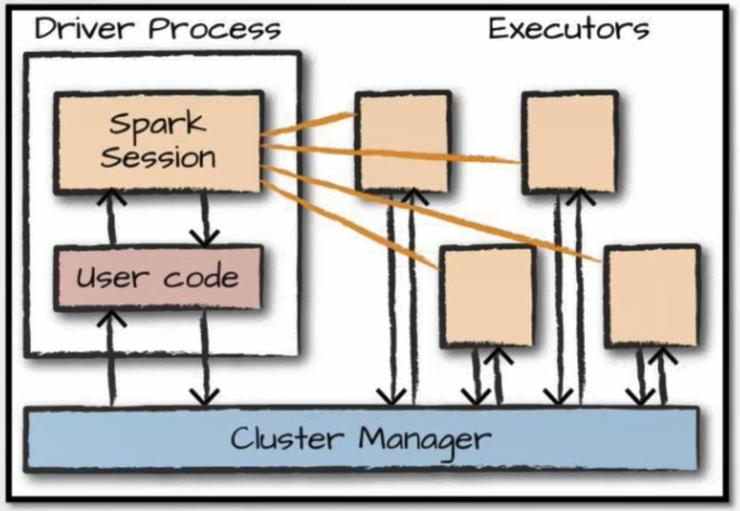


Driver Process

the driver runs the main function, sits on a node in the cluster, and is responsible for

- maintaining information about the spark application
- responding to a user's program input
- analyze, distribute and schedule work across the executors

each language API maintains the same core concepts that we described earlier. there is a spark session object available to the user, which is the entrance point to running spark code. when using spark from python or r, you don't write explicit JVM instructions; instead, you write python and r code that spark translates into code that it then can run on the executor JVMs.



Executors

responsible for carrying out the work that the driver assigns them

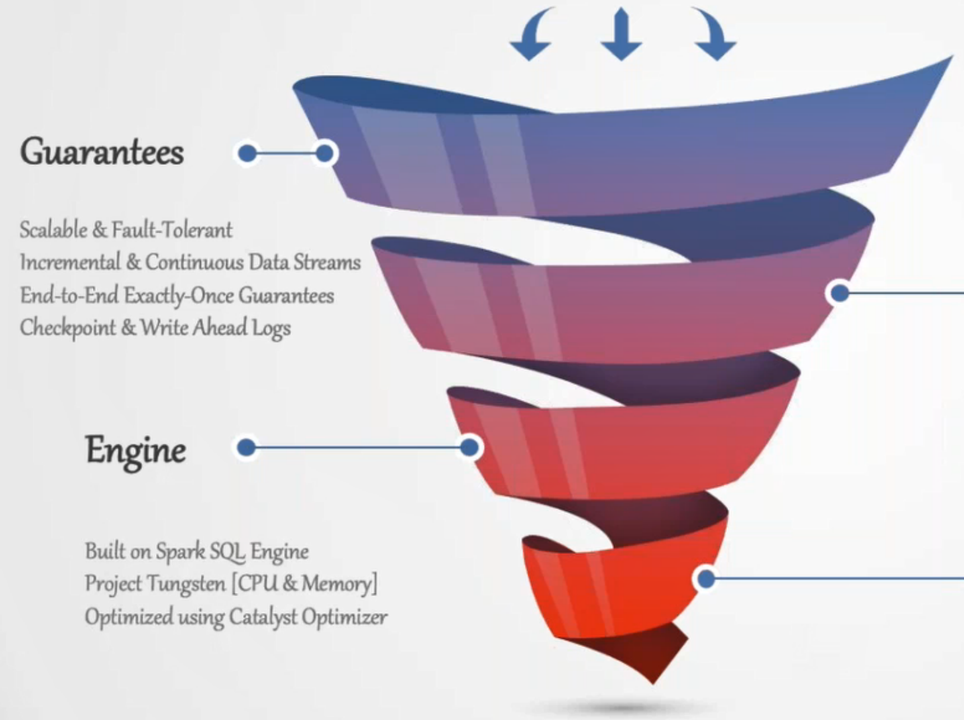
- executing code assigned by the driver
- reporting the state of the computation

RDD | DataFrame & DataSet [Comparison]

Apache Spark [Structured Streaming]

Philosophy [Moto]

Treat Data Streams = Unbounded Tables
Incremental Query over Streams



Guarantees

Scalable & Fault-Tolerant
Incremental & Continuous Data Streams
End-to-End Exactly-Once Guarantees
Checkpoint & Write Ahead Logs

Seamless

Express Streaming Computation = Batch Computation

Engine

Built on Spark SQL Engine
Project Tungsten [CPU & Memory]
Optimized using Catalyst Optimizer

Processing

Uses Micro-Batch Processing Engine
100 milliseconds with Exactly-Once Latency

Spark ML [MLlib]

MLlib – Apache Spark's Scalable Machine Learning Library

RDD-Based API vs. DataFrame-Based API

MLlib RDD-Based API [**spark.mllib**] Package [Maintenance Mode]
DataFrame-Based API [**spark.ml**] Package

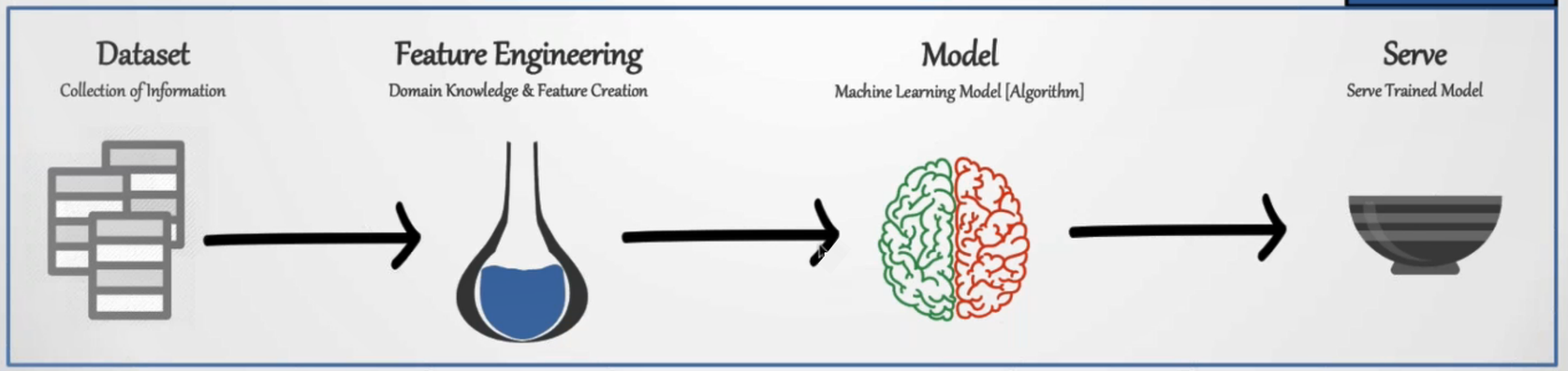
DataFrame-Based API [Spark-ML]

- User-Friendly API
- Data Sources
- SQL & DataFrame Queries
- Tungsten & Catalyst Optimizations
- Facilitate Practical ML Pipelines [Feature Transformation]

ML Library [MLlib]

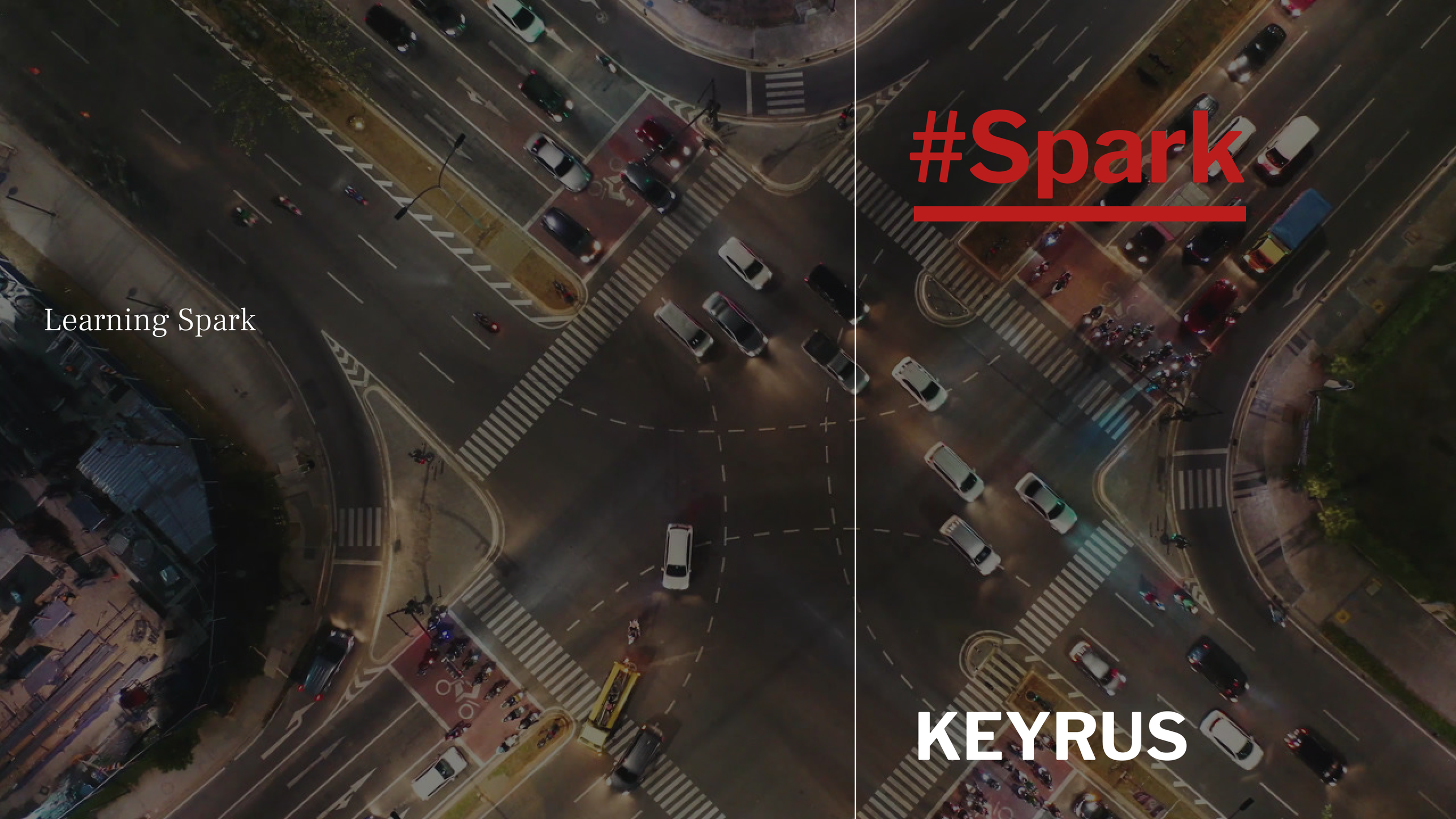
ML Algorithms – Classification | Regression | Clustering & Collaborative Filtering
Featurization – Extraction | Transformation | Dimensionality Reduction & Selection
Pipelines – Constructing | Evaluating & Tuning ML Pipelines
Persistence – Saving & Loading Algorithms, Models & Pipelines
Utilities – Linear Algebra | Statistics & Data Handling

ML Pipeline



RDD | DataFrame & DataSet [Comparison]

Develop PySpark Program for Apache Spark on [HDInsight]


Learning Spark
#Spark
KEYRUS

Conteúdo Programático

- DATALAKE
- SPARKSQL
- DATAFRAME
- EXECUTANDO SPARK EM UM CLUSTER
- DATABRICKS

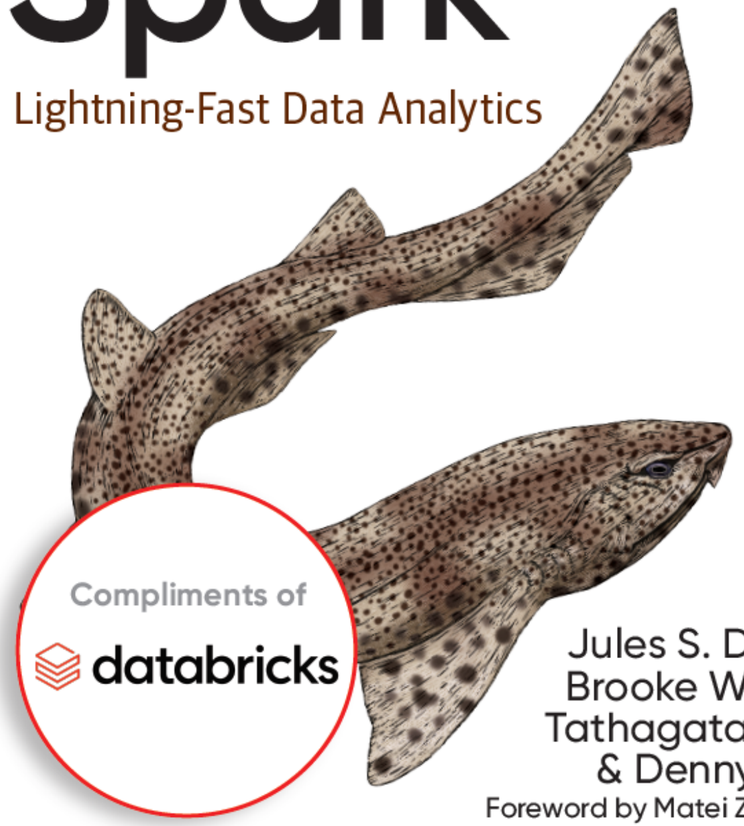
DATA ENGINEERING

O'REILLY®

2nd Edition
Covers
Apache Spark 3.0

Learning Spark

Lightning-Fast Data Analytics



Compliments of

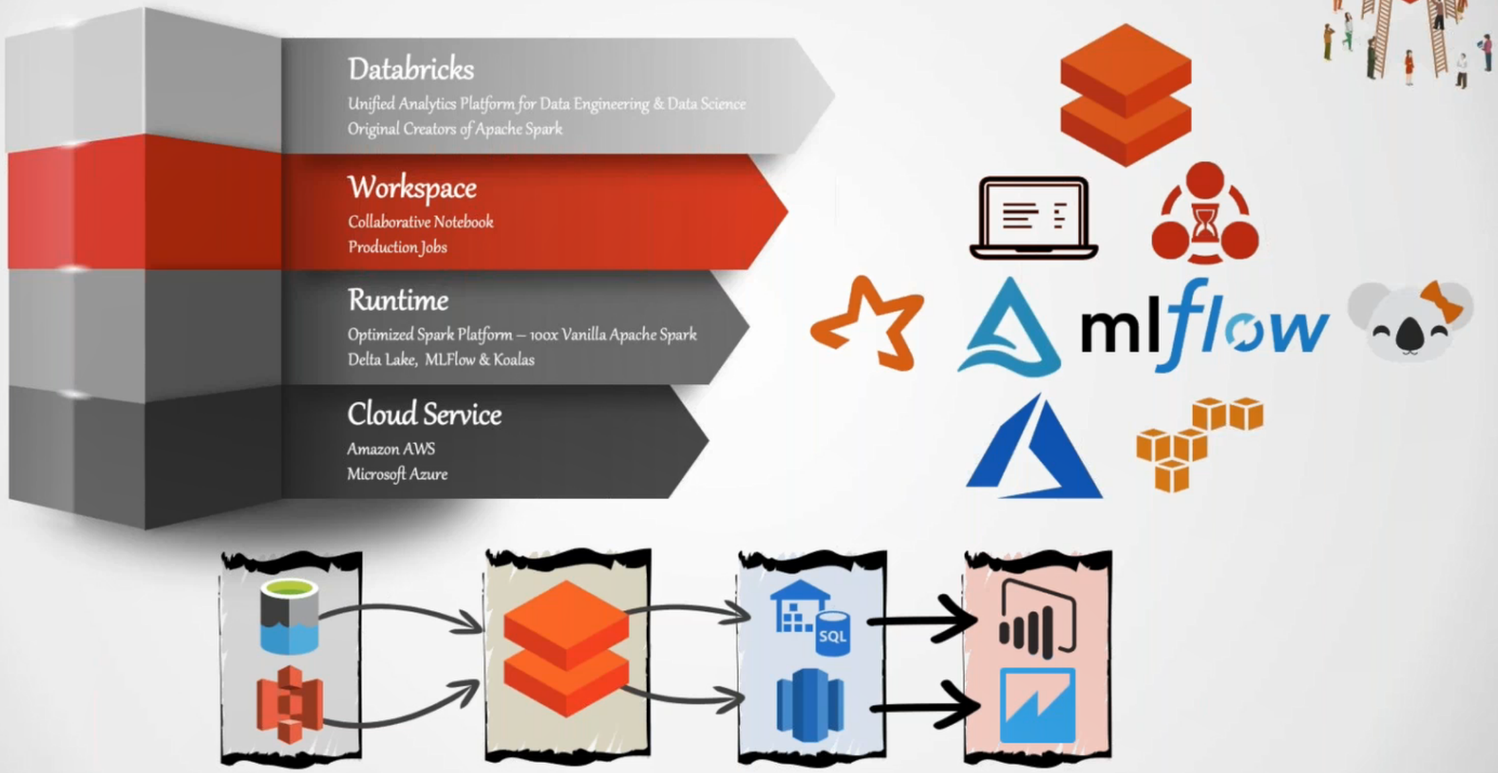
databricks

Jules S. Damji,
Brooke Wenig,
Tathagata Das
& Denny Lee

Foreword by Matei Zaharia

Using the Scala or PySpark Shell on [HDInsight]

Databricks [Overview]



Databricks [TimeLine]

SPARK+AI SUMMIT EUROPE

2013 — Databricks Founded
Creators of Apache Spark
Raise $14M in Series A Funding

2014 — Databricks Cloud Announced
Amazon AWS Only
Raise $33M in Series B Funding

2015 — Databricks Cloud Publicly Available
Amazon AWS

2016 — Databricks Debuts Free Community Edition
Raise $60M in Series C Funding

2017 — Microsoft Announce Azure Databricks
Raise $140M in Series D Funding

2018 — Microsoft Announce GA of
Azure Databricks

2019

Getting Started with Databricks

Lambda Architecture

Batch-Layer

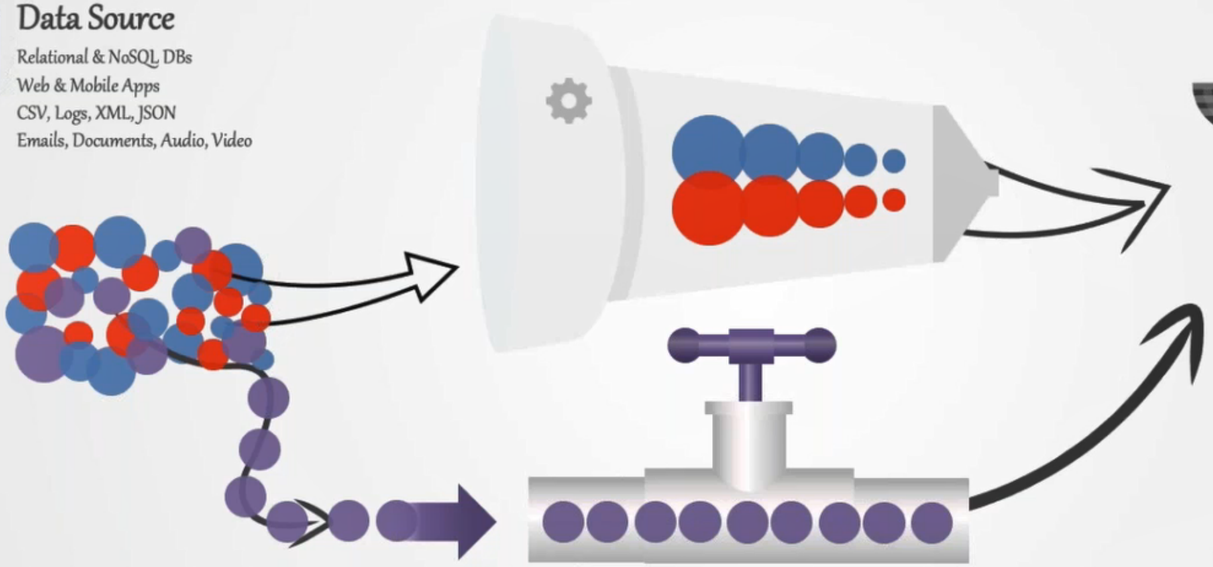
Data Storage - Data Stored in a Data Lake
Batch-Processing - New Data is Computed & Processed

Nathan Marz

Coined in 2011
Founder of Red Planet Labs
Software Engineer at Twitter

Data Source

Relational & NoSQL DBs
Web & Mobile Apps
CSV, Logs, XML, JSON
Emails, Documents, Audio, Video



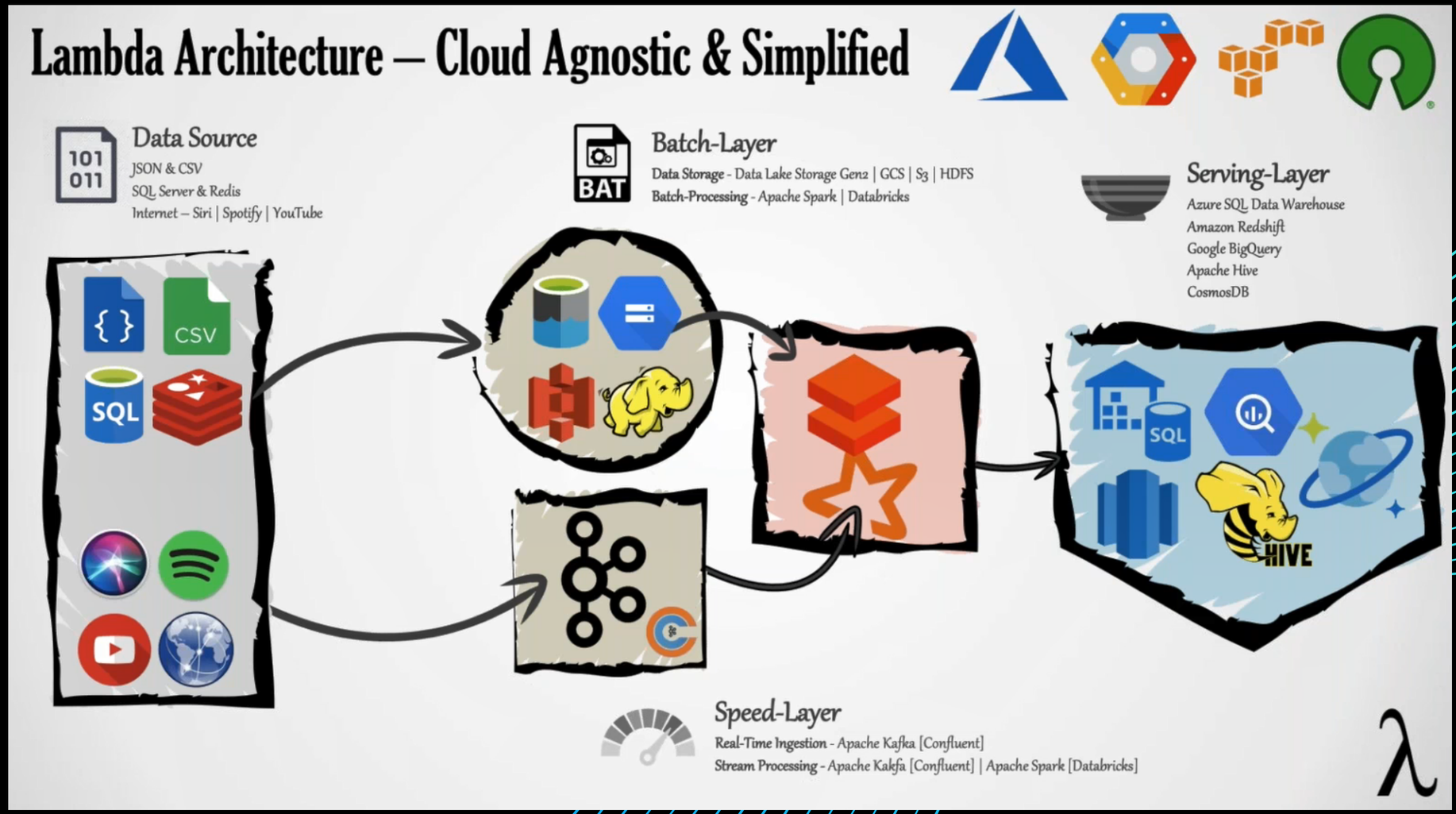
Serving-Layer

Output of Batch & Speed Layer
Processed & Computed Data
Ad-Hoc Queries & Data Analysis

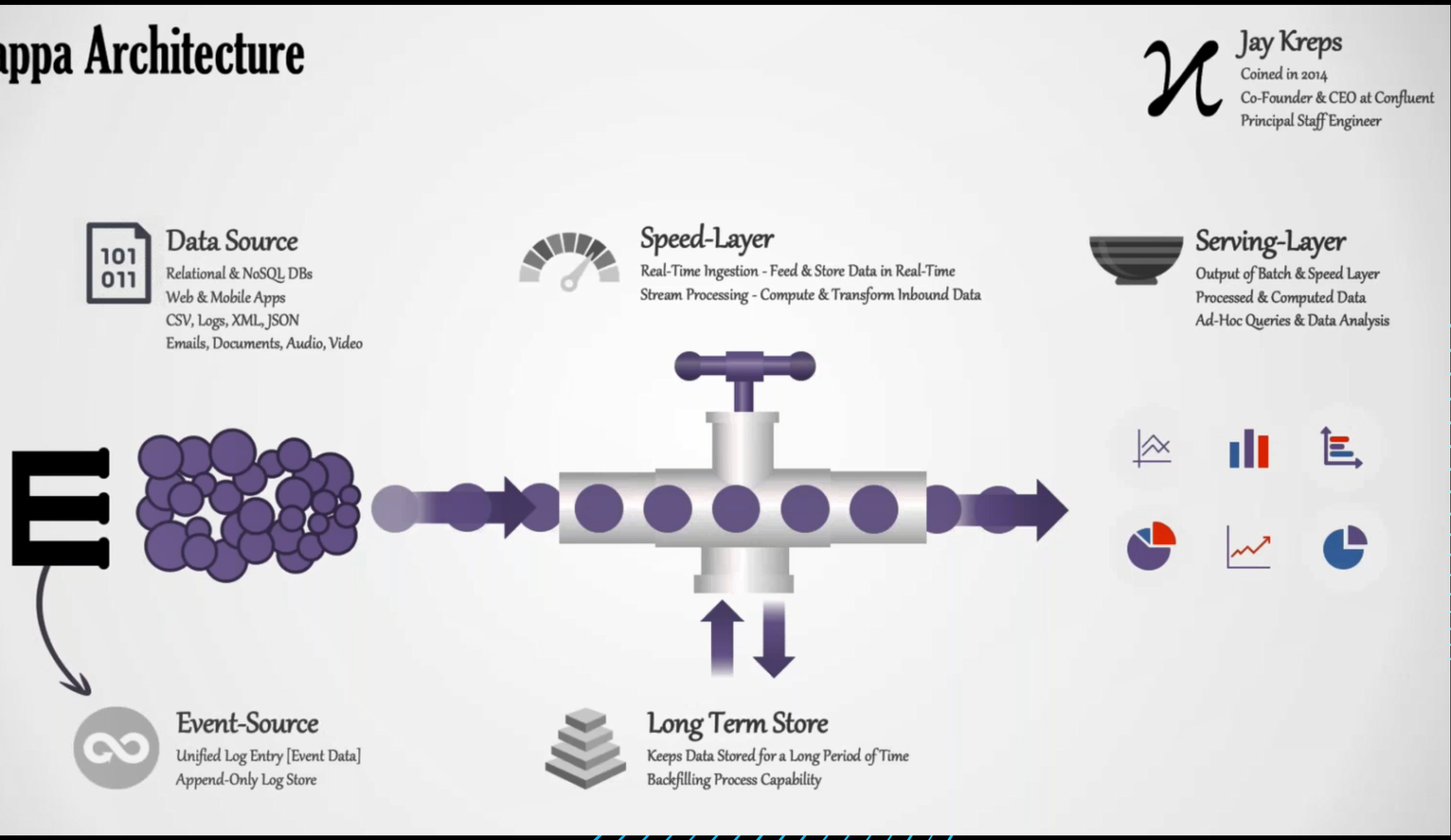
Speed-Layer

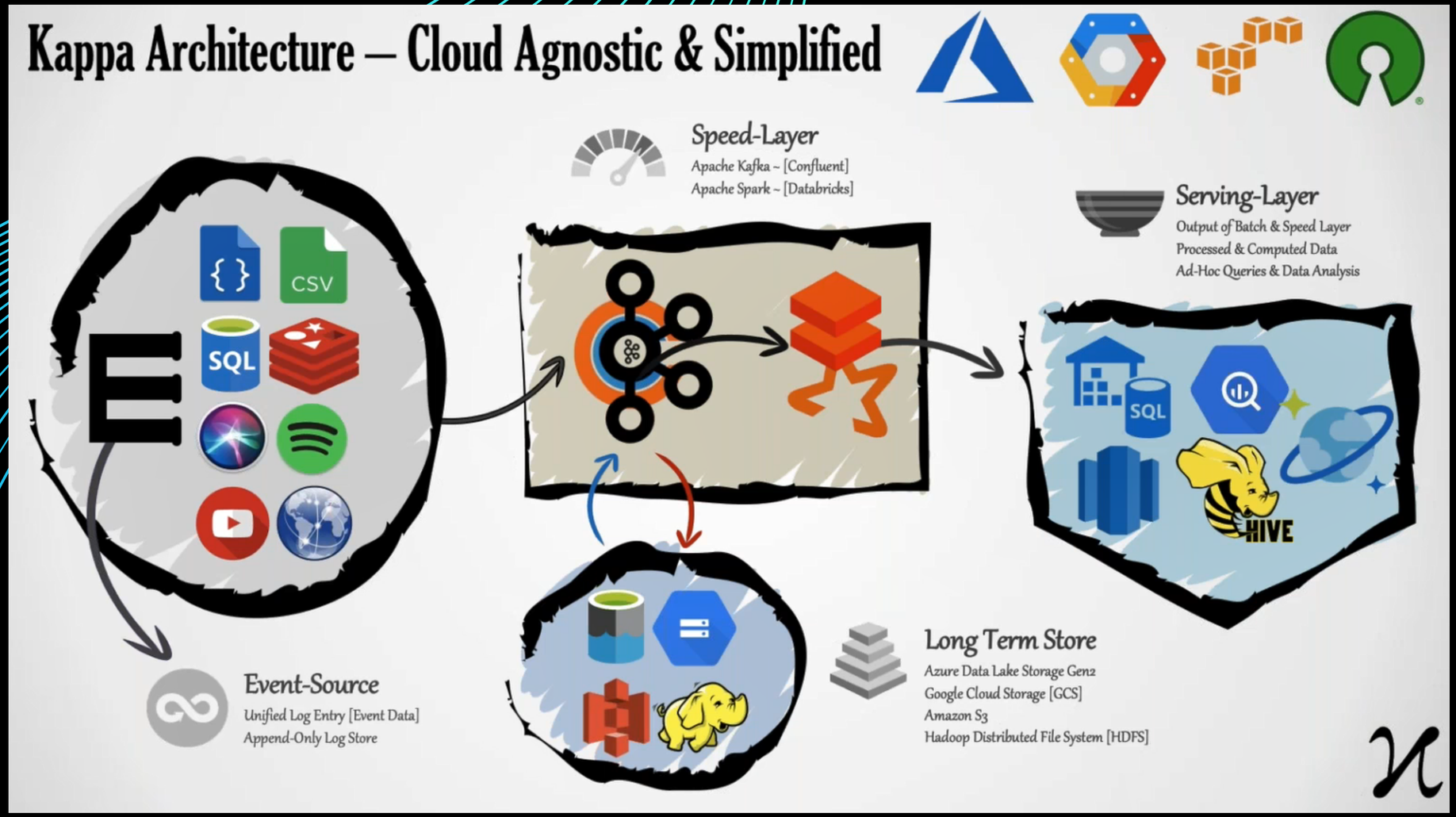
Real-Time Ingestion - Feed & Store Data in Real-Time
Stream Processing - Compute & Transform Inbound Data

Lambda Architecture – Cloud Agnostic & Simplified

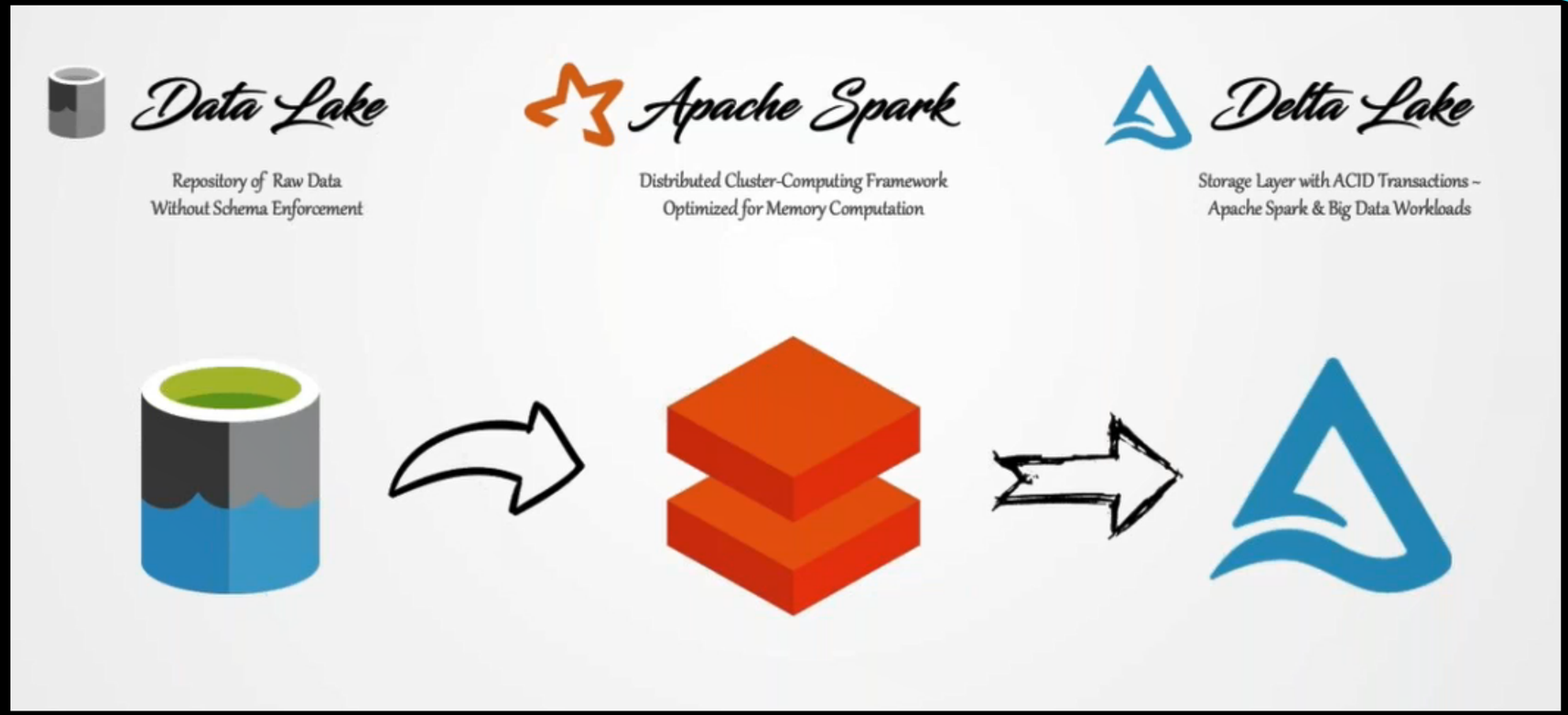


Kappa Architecture

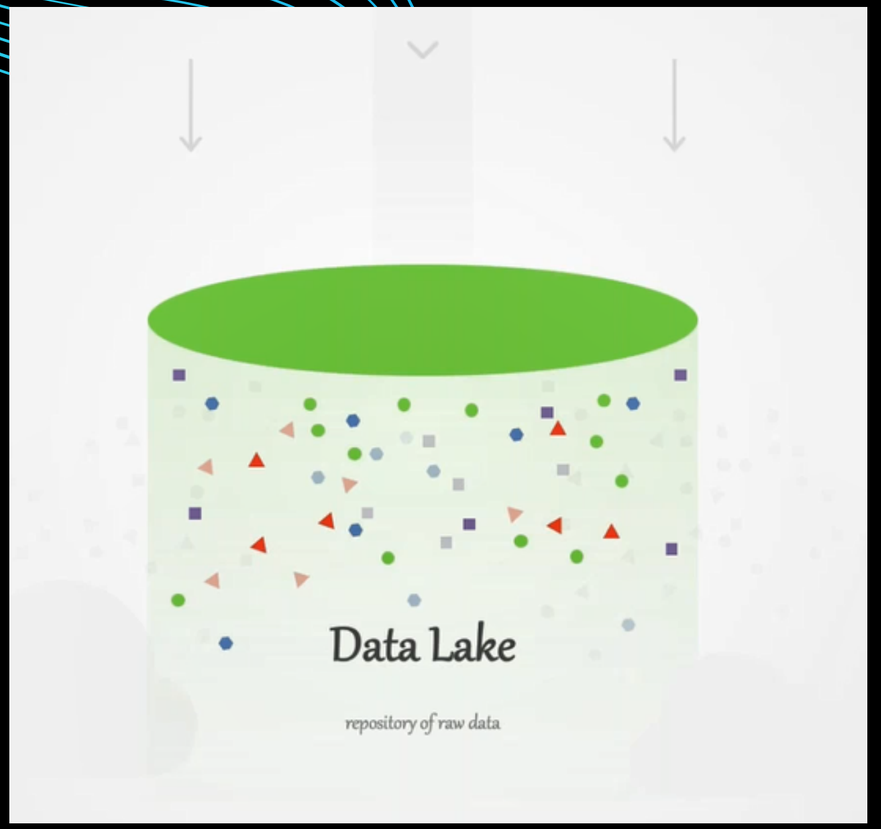



Kappa Architecture – Cloud Agnostic & Simplified
Speed-Layer
Apache Kafka ~ [Confluent]
Apache Spark ~ [Databricks]
Serving-Layer
Output of Batch & Speed Layer
Processed & Computed Data
Ad-Hoc Queries & Data Analysis
CSV
SQL
SQL
HIVE
Event-Source
Unified Log Entry [Event Data]
Append-Only Log Store
Long Term Store
Azure Data Lake Storage Gen2
Google Cloud Storage [GCS]
Amazon S3
Hadoop Distributed File System [HDFS]

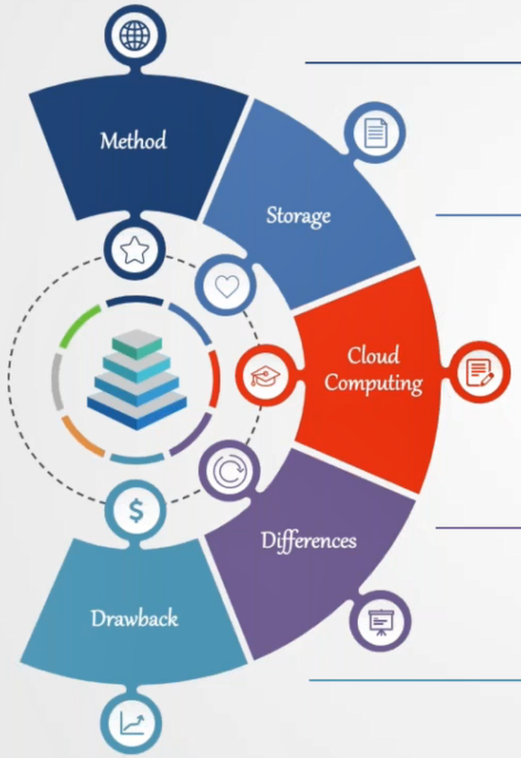
Use Case ~ Batch-ETL




RDBMS
SQL Server | Oracle | PostgreSQL | MySQL
NoSQL
MongoDB | Cassandra | Redis
Web
Web App | Mobile App
Files
CSV | XML | JSON
Process


Data Lake
repository of raw data

Data Lake [Concepts]



Repository of Raw Data [Data Democratization]
Data Lake – James Dixon in 2010
Democratization of Data – [Unsiloed Data]

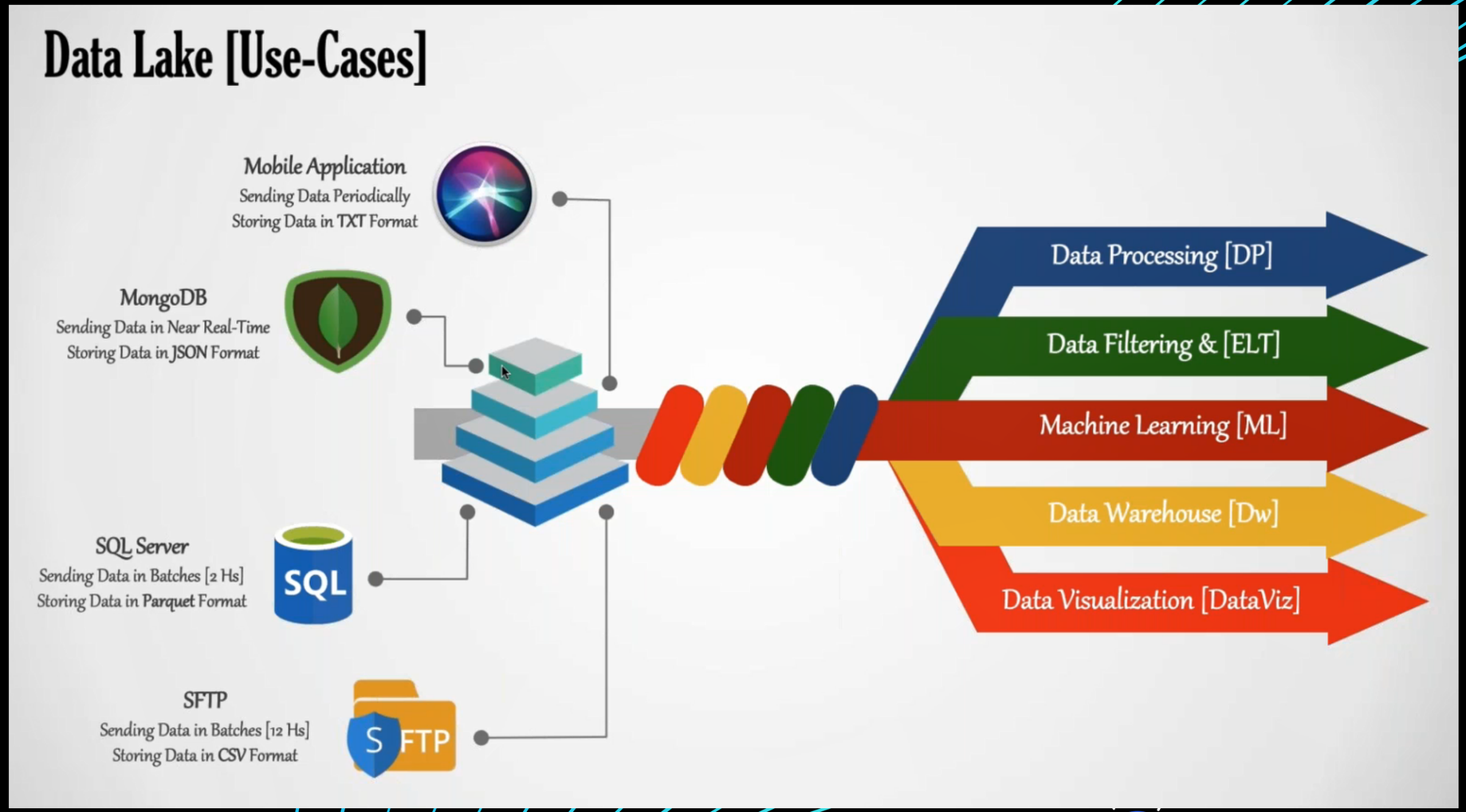
Structured – Relational Databases
Semi-Unstructured – CSV, Logs, XML & JSON
Unstructured – Emails, Documents, Binaries, Audio & Video

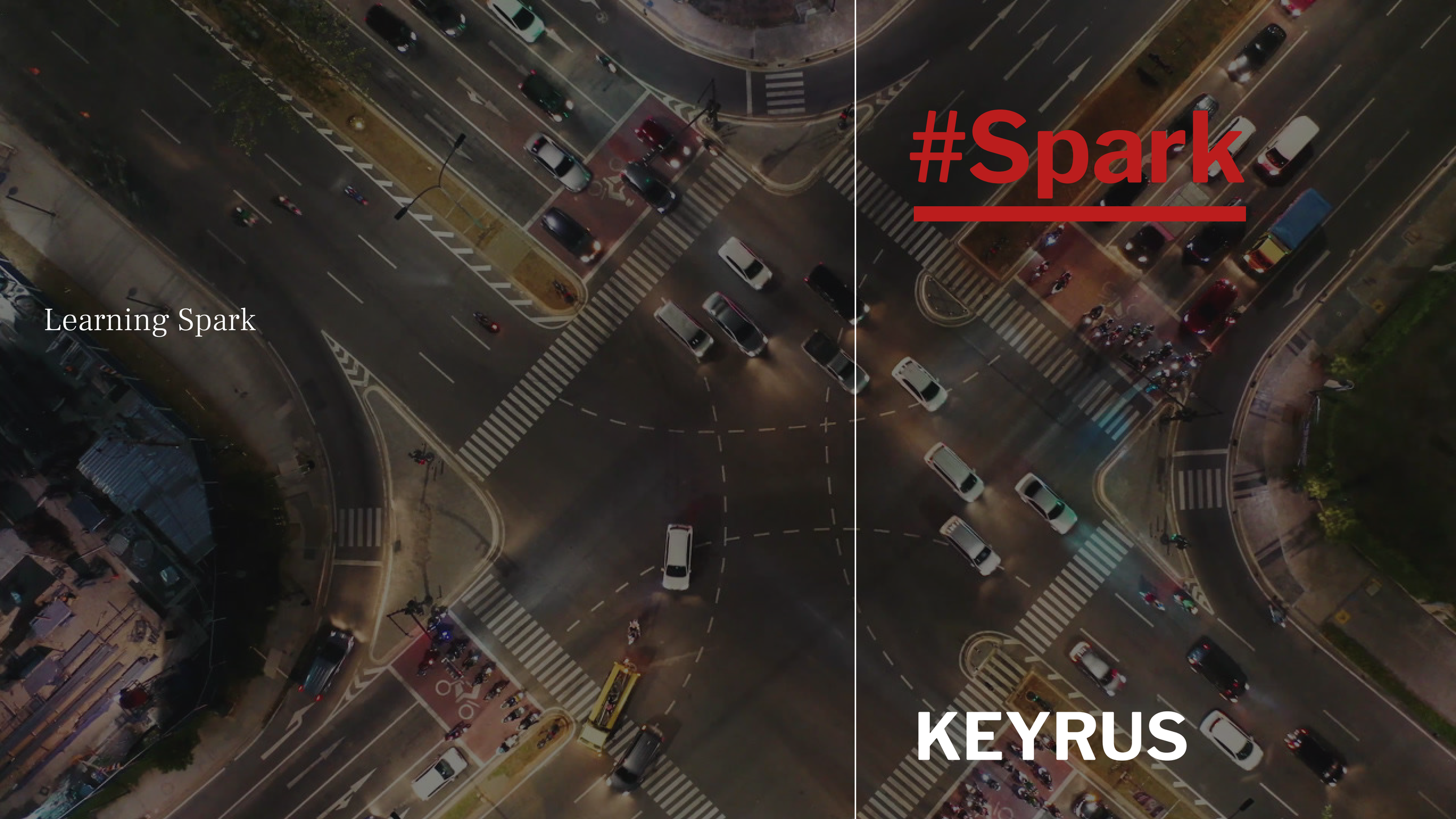
Azure – Azure Blob Storage & Azure Data Lake Storage Gen2
AWS – Amazon S3 & AWS Lake Formation
GCP – Google Cloud Storage

Data Mart – Subset of Decision Support for Departments
Data Warehouse – Decision Making Support for Enterprise-Level
Data Lake – Raw Data [Source of Truth]

Data Swamp – Highly Disorganized Data
Data Governance – Management & Control of Data
Data Quality – Accuracy and Data Veracity
Data Security – Protecting Digital Data

Data Lake [Use-Cases]



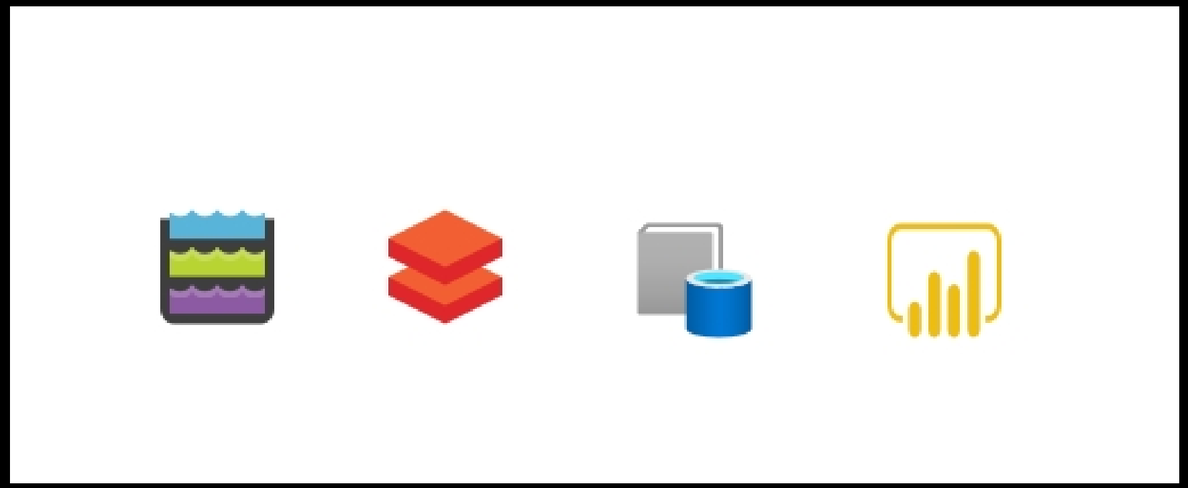

Learning Spark
#Spark
KEYRUS

Conteúdo Programático

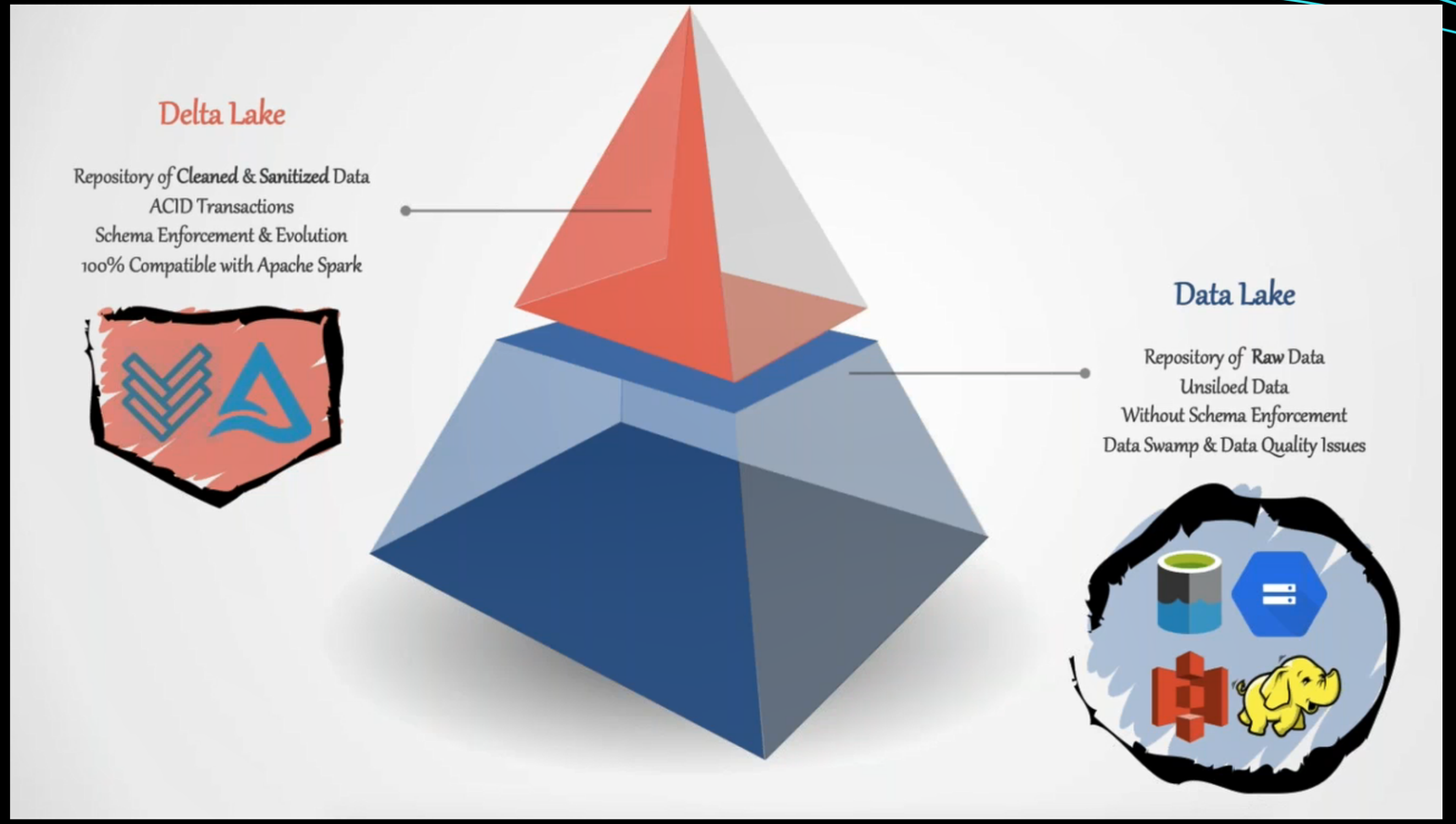
- DATA LAKE
- DELTA LAKE
- DESIGN PATTERNS
- EXECUTANDO SPARK EM UM CLUSTER
- DEMO ON DATABRICKS

DATA ENGINEERING

Best Practices Organizing a Data Lake Repository



Data Lake vs. Delta Lake



Delta Lake [Features]

Delta Lake

Is an Open-Source Storage Layer – ACID Transactions to Apache Spark & Big Data Workloads

unified analytics engine for big data

storage layer [format] with acid capabilities

most common data lakes

ACID Transactions

data lakes typically have multiple data pipelines reading and writing data concurrently, and data engineers must go through a tedious process to ensure data integrity, due to the lack of transactions. it provides serializability, the strongest level of isolation level.

Open Format

all data in delta lake is stored in apache parquet format enabling delta lake to leverage the efficient compression and encoding schemes that are native to parquet.

Audit History

delta lake transaction log records details about every change made to data providing a full audit trail of the changes.

Scalable Metadata Handling

delta lake treats metadata just like data, leveraging spark's distributed processing power to handle all its metadata. as a result, delta lake can handle petabyte-scale tables with billions of partitions and files at ease.

Unified Batch & Streaming

a table in delta lake is both a batch table, as well as a streaming source and sink. streaming data ingest, batch historic backfill, and interactive queries all just work out of the box.

Updates & Deletes

delta lake supports Scala, Java, Python, SQL APIs to merge, update and delete datasets. this allows you to easily comply with GDPR and CCPA and simplifies use cases like change data capture.

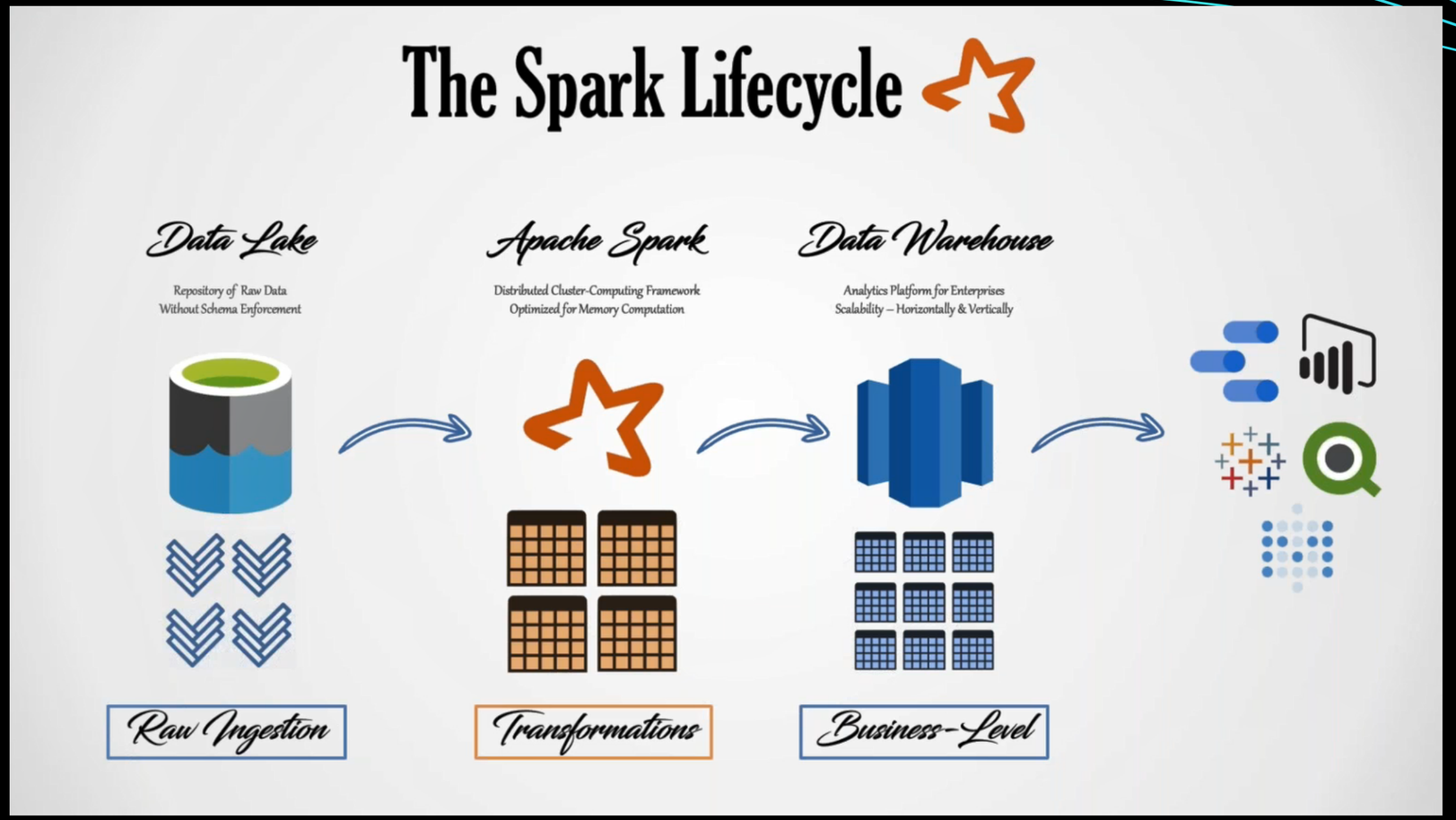
Time Travel [Data Versioning]

delta lake provides snapshots of data enabling developers to access and revert to earlier versions of data for audits, rollbacks or to reproduce experiments.

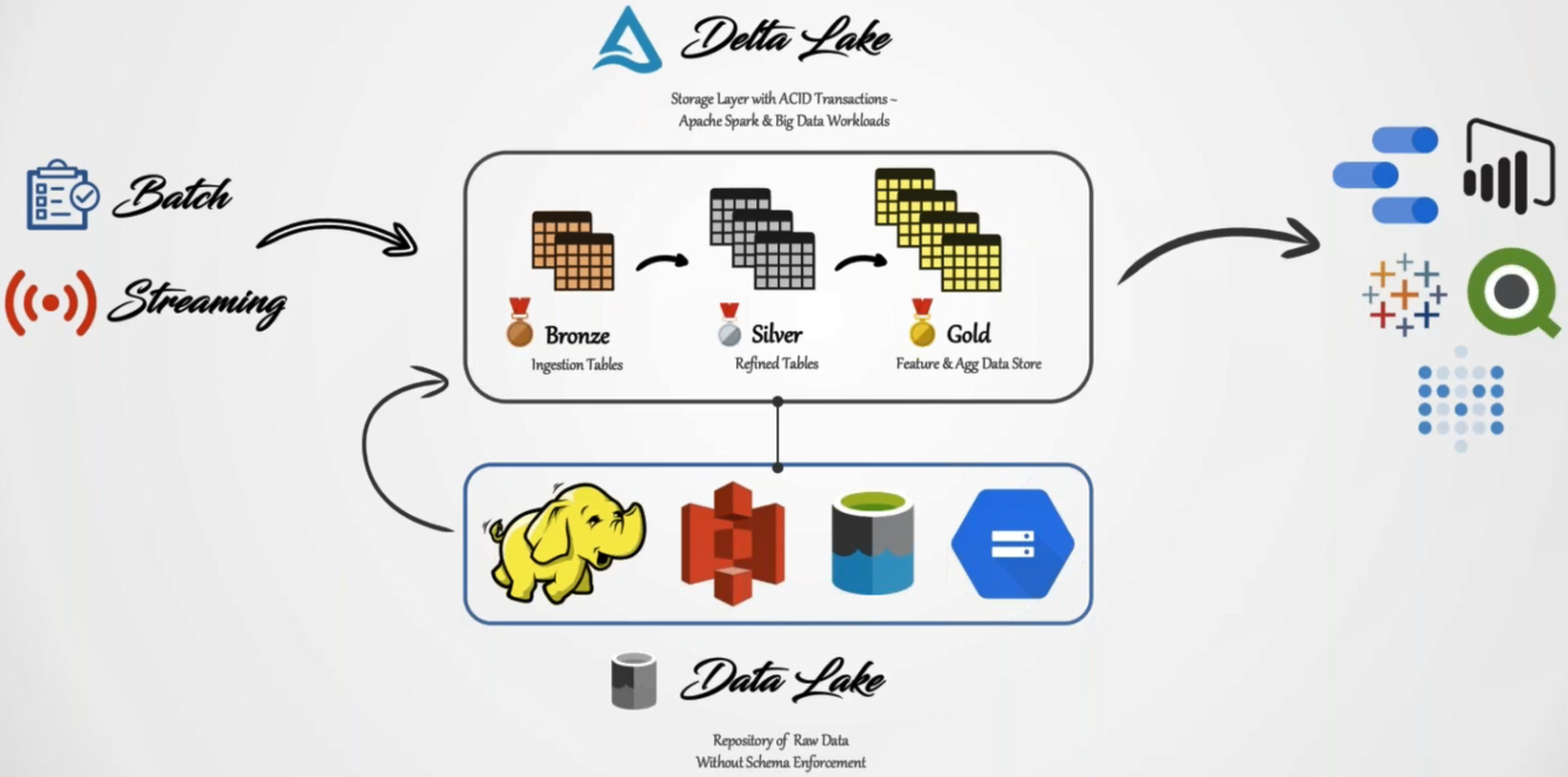
Schema Enforcement & Evolution

delta lake provides the ability to specify your schema and enforce it. this helps ensure that the data types are correct and required columns are present, preventing bad data from causing data corruption. delta lake enables you to make changes to a table schema that can be applied automatically, without the need for cumbersome dll.

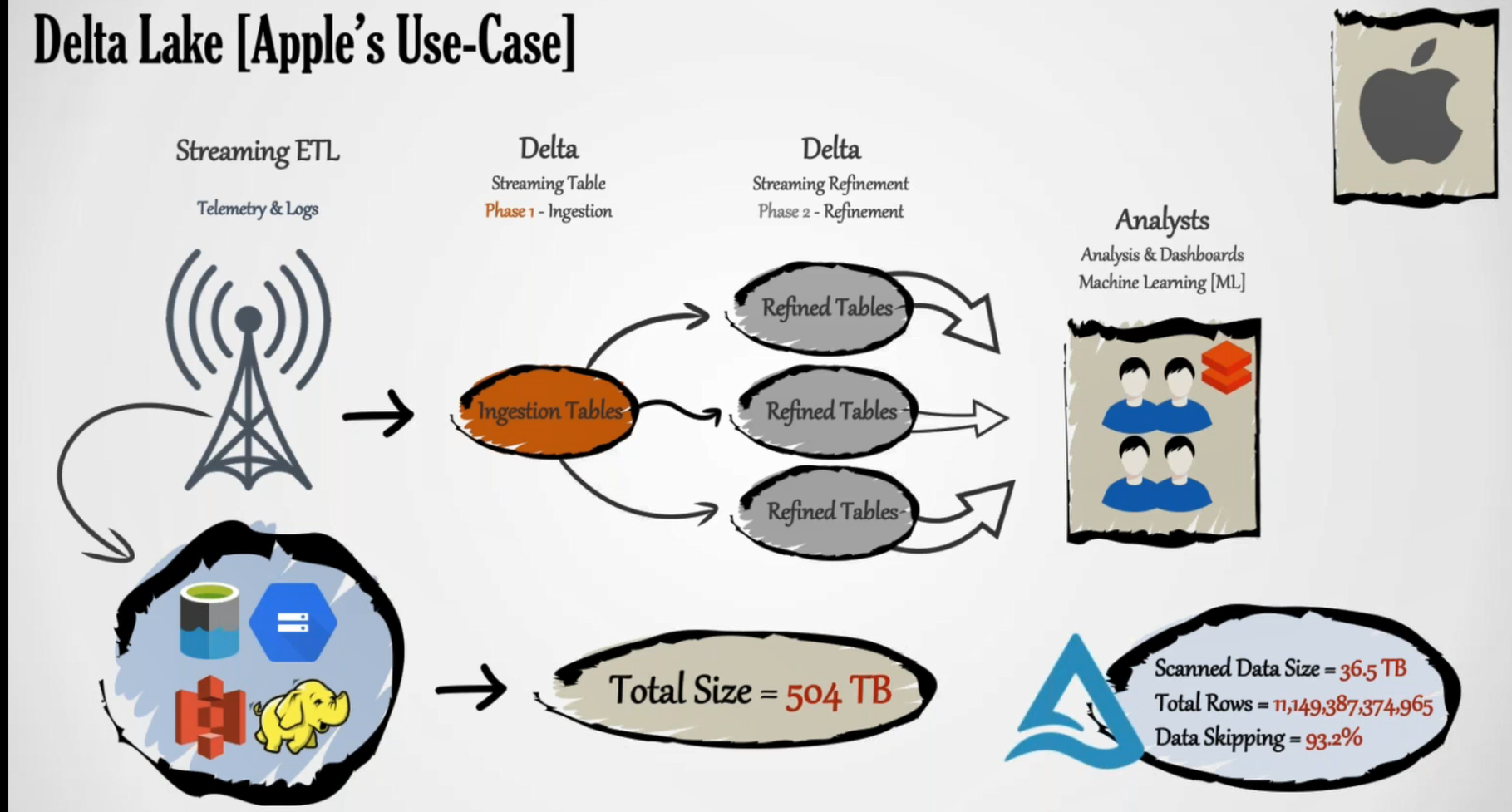
Data Lake vs. Delta Lake



The Delta Architecture



Delta Lake [Apple's Use-Case]



Data Lakehouse

Best of a Data Warehouse and Data Lake

Data Warehouse [Dw]

Decision Support for BI Applications
Since 1980s evolving & MPP Architecture
Structured Data & No Cost Efficient

Data Lake [DL]

Large Amount of Data [DV] from Different Sources
Since 2000s being a Data Raw Repository
Without Transactions, Data Enforcement, Lack of Consistency
Batch & Streaming Jobs not Performing as Expected
AI Advances based on Unstructured Data (Text, Images, Video, Audio)

Lakehouse [LH]

Similar Structure of Dw with a Low-Cost Storage with Features:

- Transaction Support
- Schema Enforcement and Governance
- BI Support
- Decoupled Storage & Computation
- Openness with Structured & Unstructured Data
- End-to-End Streaming

Data Ingestion into Data Lakehouse Best Practices

Network of Partners [Connectors]

essential ecosystem of connectors to bring data into Delta Lake
Azure Data Factory, Fivetran, Qlik, Infoworks, StreamSets, Syncsoft

Cloud Storage [Auto Loader]

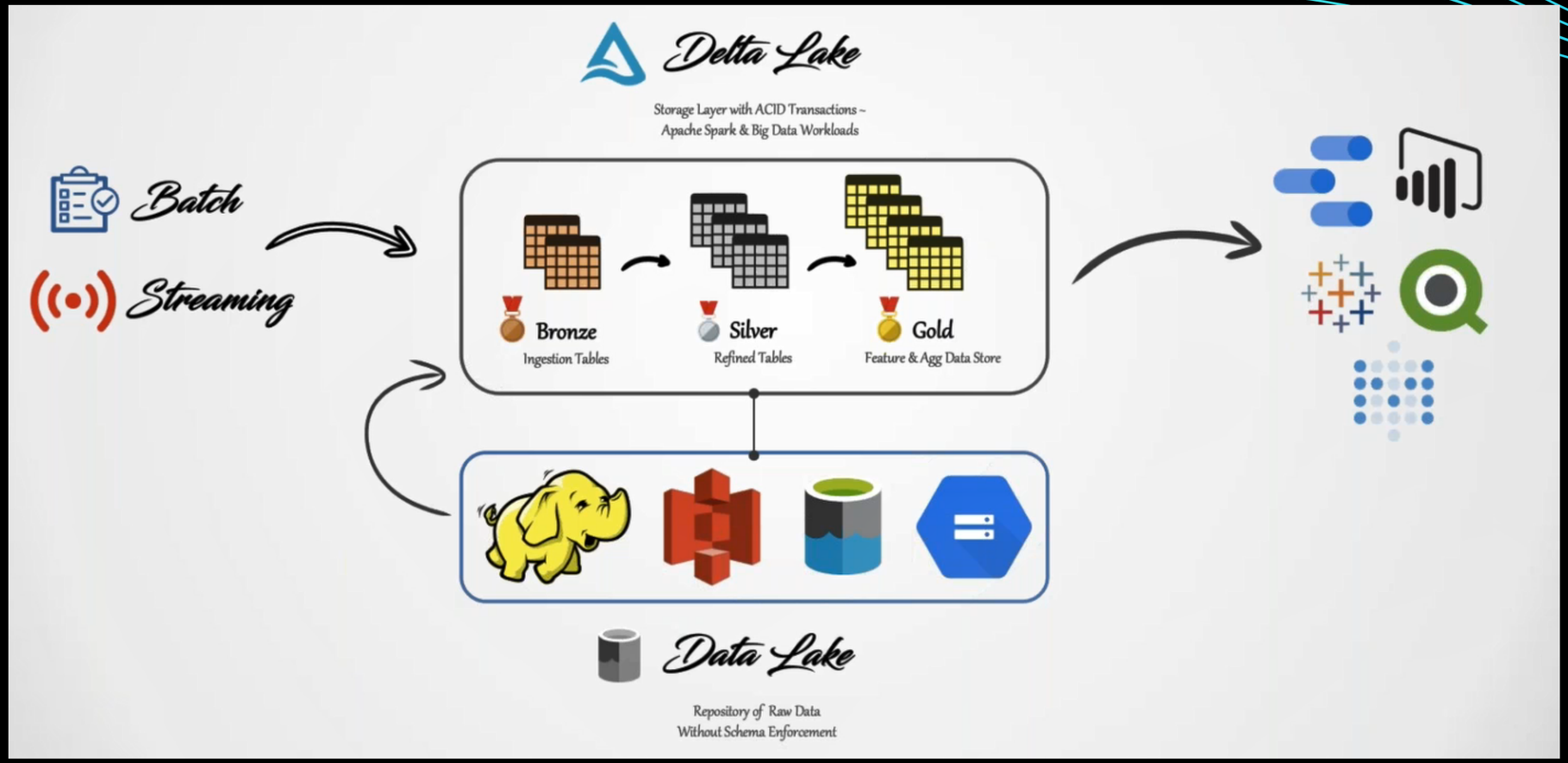
loading data continuously from cloud stores with **Exactly-Once** guarantees at low cost, low latency and minimal DevOps work. List is one of the most expensive operations. **Auto Loader** is an optimized file source that uses **Structured Streaming**

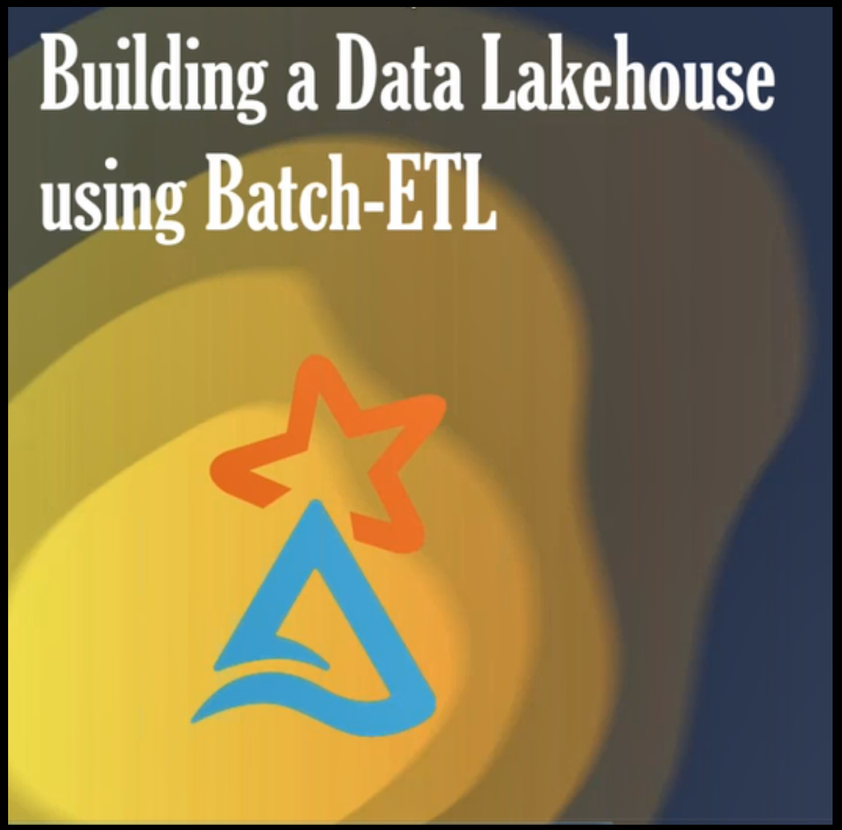
Streaming Load [Structured Streaming]

loading data continuously from **Apache Kafka** with **Exactly-Once** using the Structured Streaming API

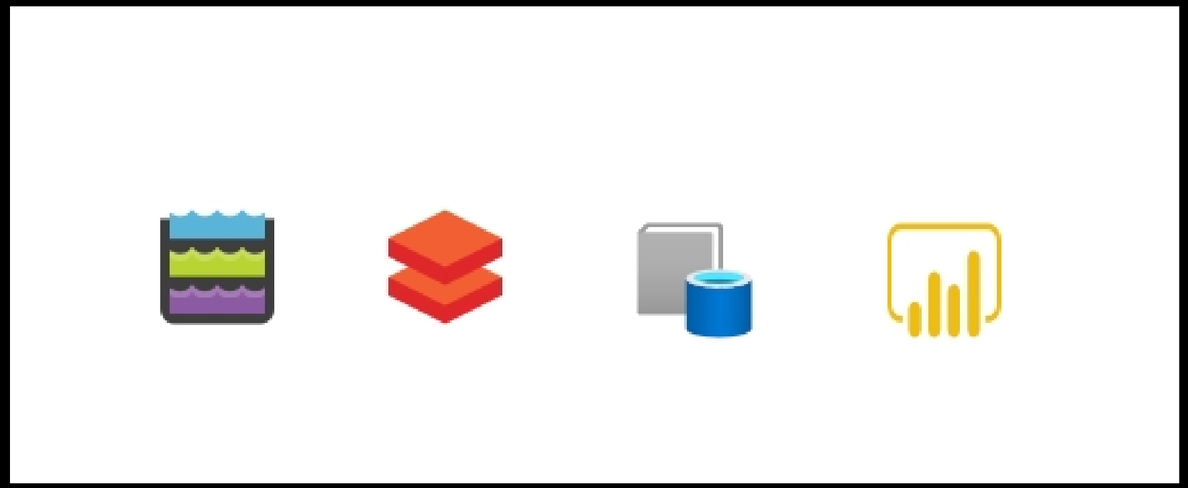
Delta Lake

storage system with ACID capabilities that is designated to build a **Data Lakehouse**


Delta Lake
Storage Layer with ACID Transactions ~
Apache Spark & Big Data Workloads
Batch
Streaming
Bronze
Ingestion Tables
Silver
Refined Tables
Gold
Feature & Agg Data Store
Data Lake
Repository of Raw Data
Without Schema Enforcement


Building a Data Lakehouse
using Batch-ETL

Best Practices Organizing a Data Lake Repository

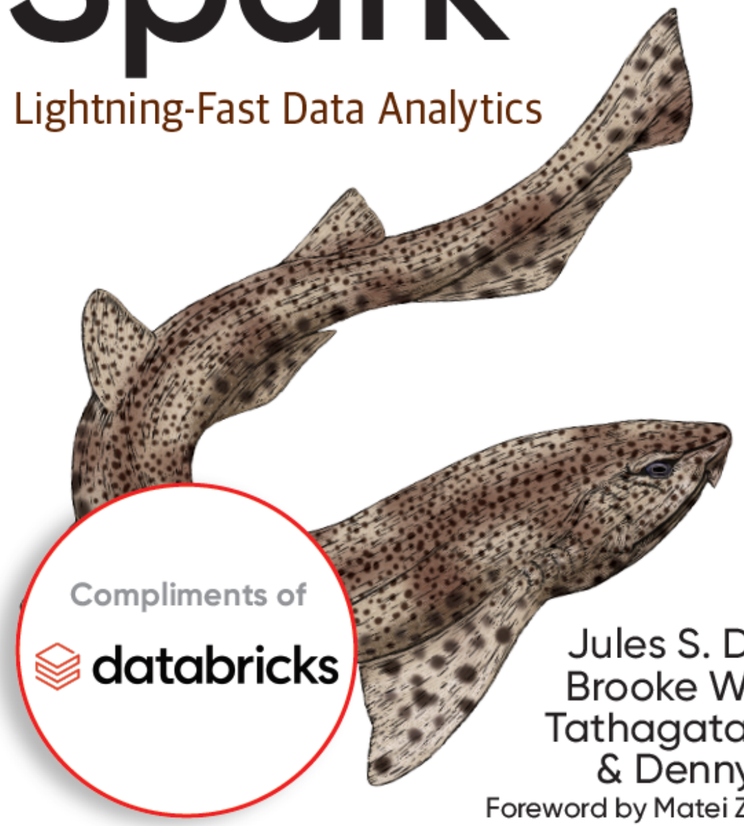


O'REILLY®

2nd Edition
Covers
Apache Spark 3.0

Learning Spark

Lightning-Fast Data Analytics



Compliments of

databricks

Jules S. Damji,
Brooke Wenig,
Tathagata Das
& Denny Lee

Foreword by Matei Zaharia