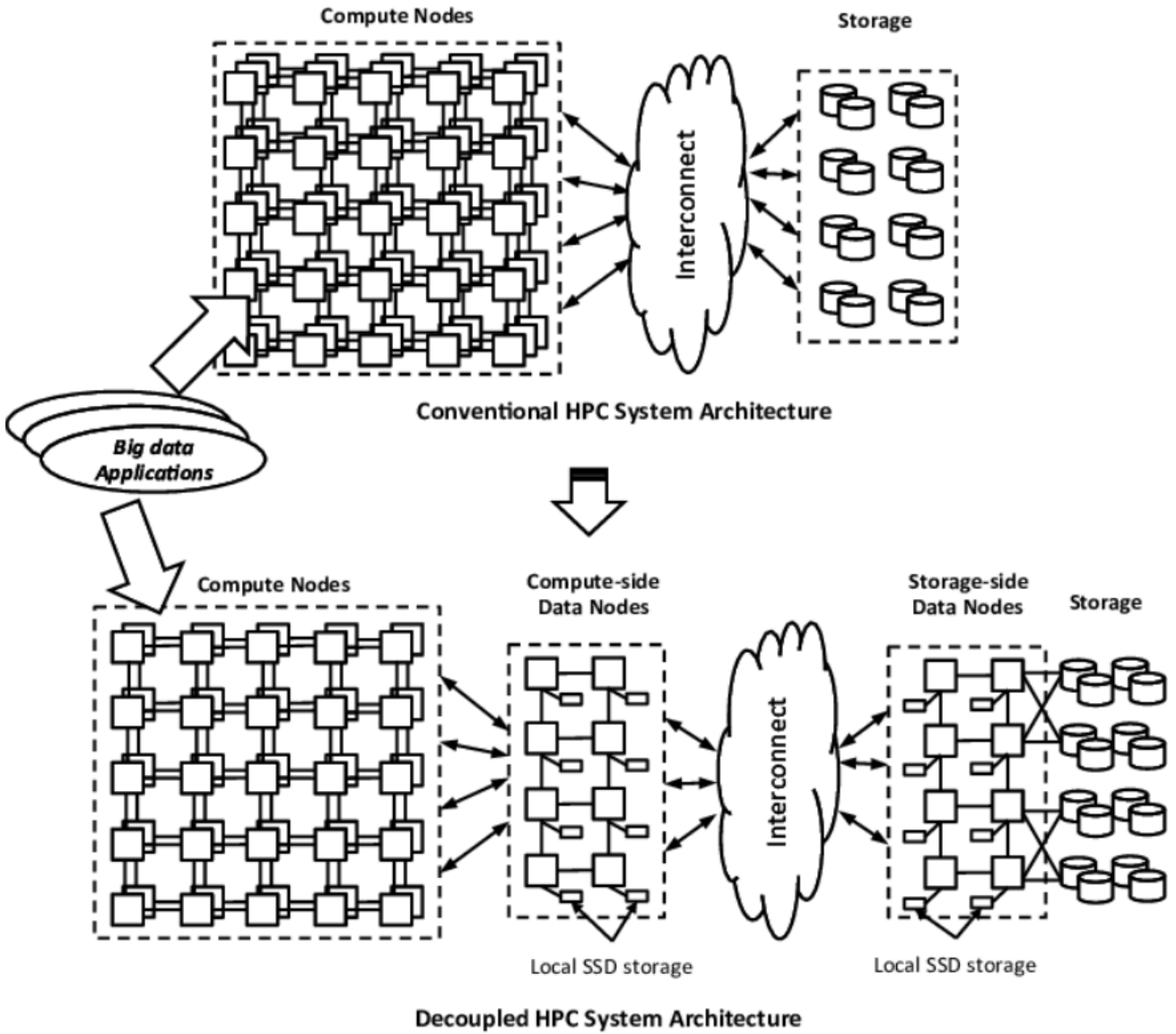
ie
SCHOOL OF
HUMAN SCIENCES
& TECHNOLOGY

OPERATING SYSTEMS & PARALLEL COMPUTING

Big Data Platform

Decoupled High Performance Computing System Architecture



ie
SCHOOL OF
HUMAN SCIENCES
& TECHNOLOGY

BigData Overview

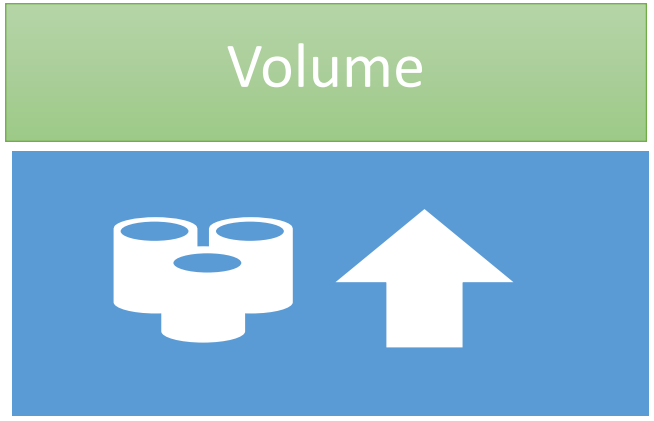
Big Data

- History
- Google
  - Started with My SQL for search engine (scalability was major issue)
  - Developed solution from scratch
    - Distributed file system – GFS
    - Distributed processing – Map Reduce
    - Big Table

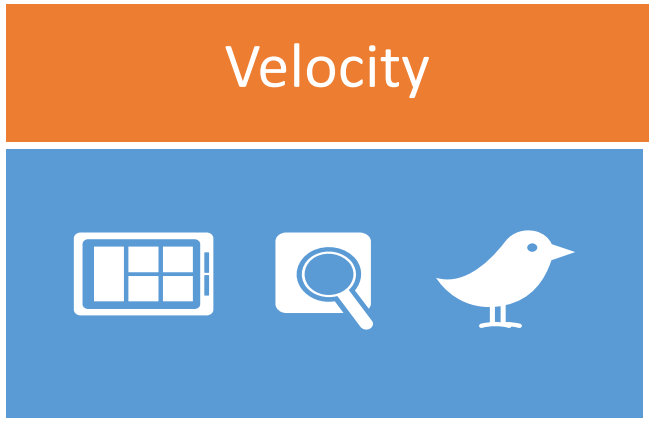
What is Big Data?

Extremely large data sets that may be analyzed computationally to reveal patterns, trends, and associations, especially relating to human behavior and interactions - Oxford Dictionary

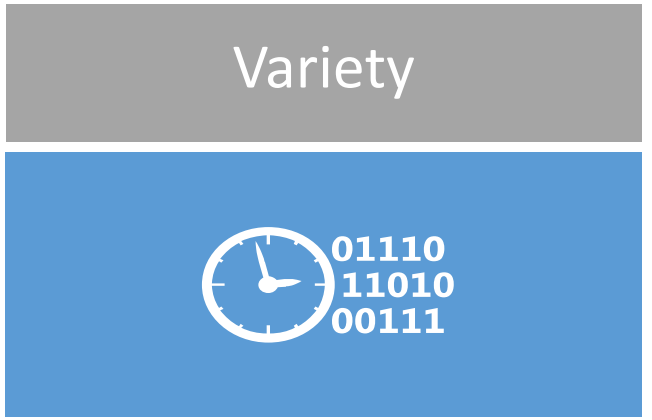
When do I need Big Data?



Huge amounts of data

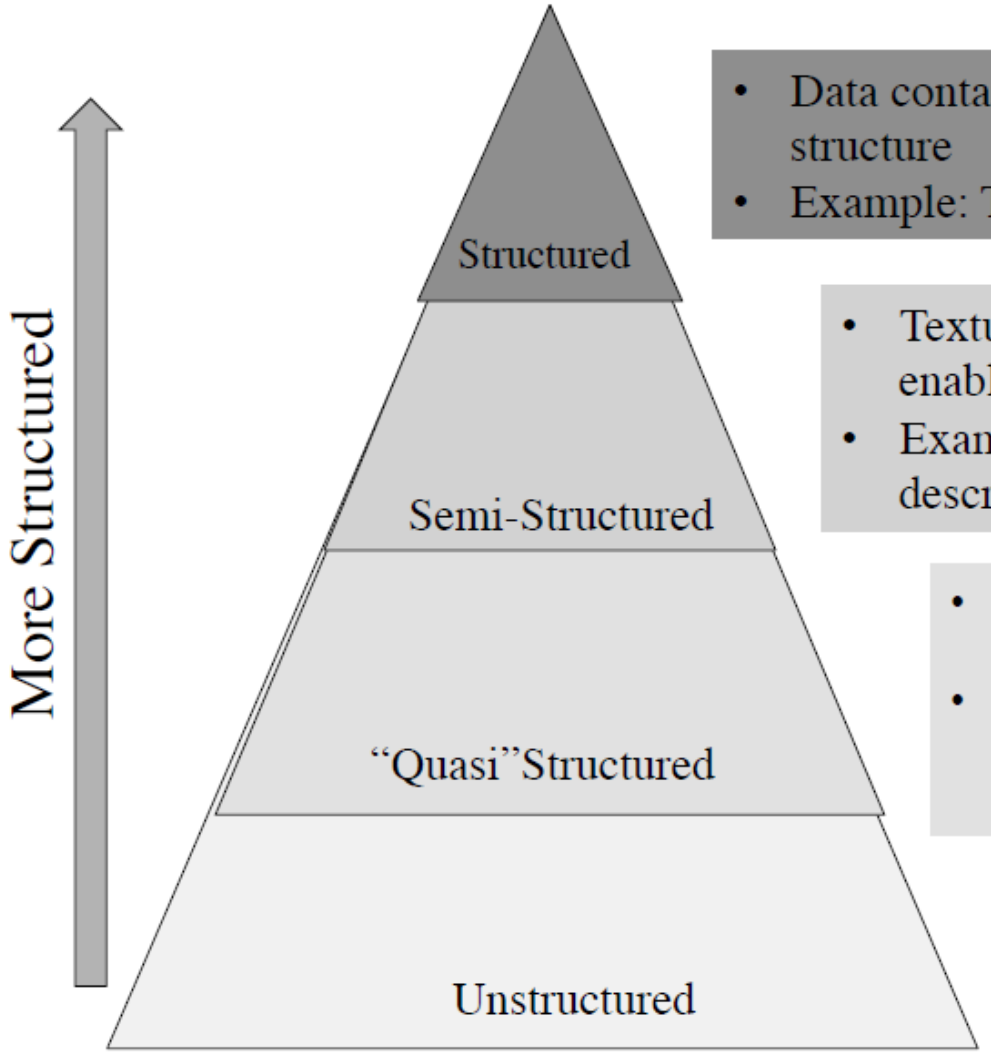


Speed at which data needs to be processed



Structured and Semi Structured

Big Data Characteristics: Data Structures



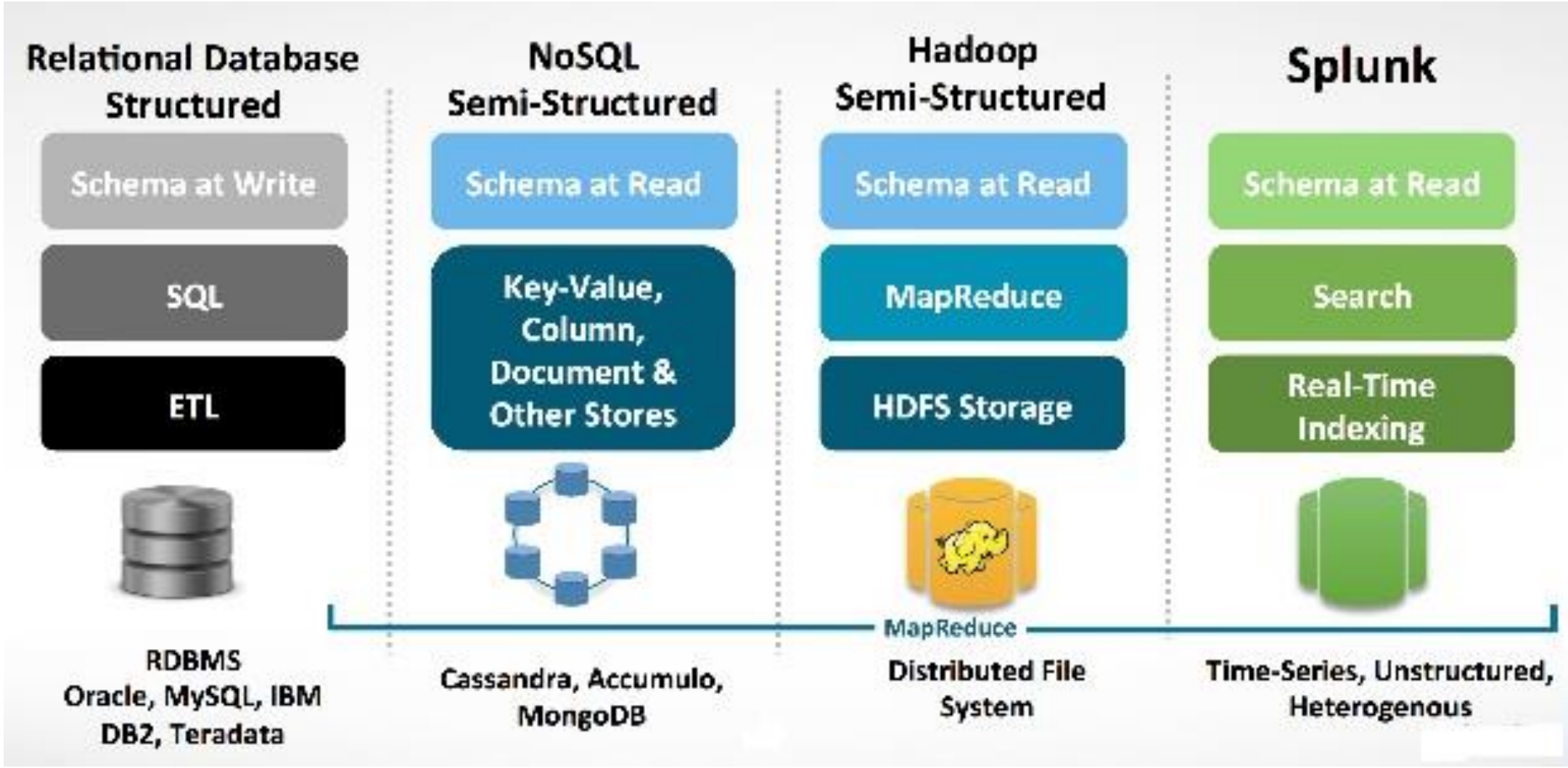
- Data containing a defined data type, format, structure
- Example: Transaction data in OLTP and OLAP

- Textual data with discernable pattern, enabling parsing
- Example: XML data files that are self describing by xml schema

- Textual data with erratic data format, can be formatted with effort tools and time
- Example, web clickstream data that may have some inconsistencies in data values and formats

- Data that has no inherent structure and is stored as different types of files
- Text documents, PDFs, images, video

Big Data Technologies



Why Hadoop is Invented?

- Storage for Large Datasets
  - The conventional RDBMS is incapable of storing huge amounts of Data. The cost of data storage in available RDBMS is very high. As it incurs the cost of hardware and software both.
- Handling data in different formats
  - The RDBMS is capable of storing and manipulating data in a structured format. But in the real world we have to deal with data in a structured, unstructured and semi-structured format.
- Data getting generated with high speed:
  - The data in oozing out in the order of tera to peta bytes daily. Hence we need a system to process data in real-time within a few seconds. The traditional RDBMS fail to provide real-time processing at great speeds.

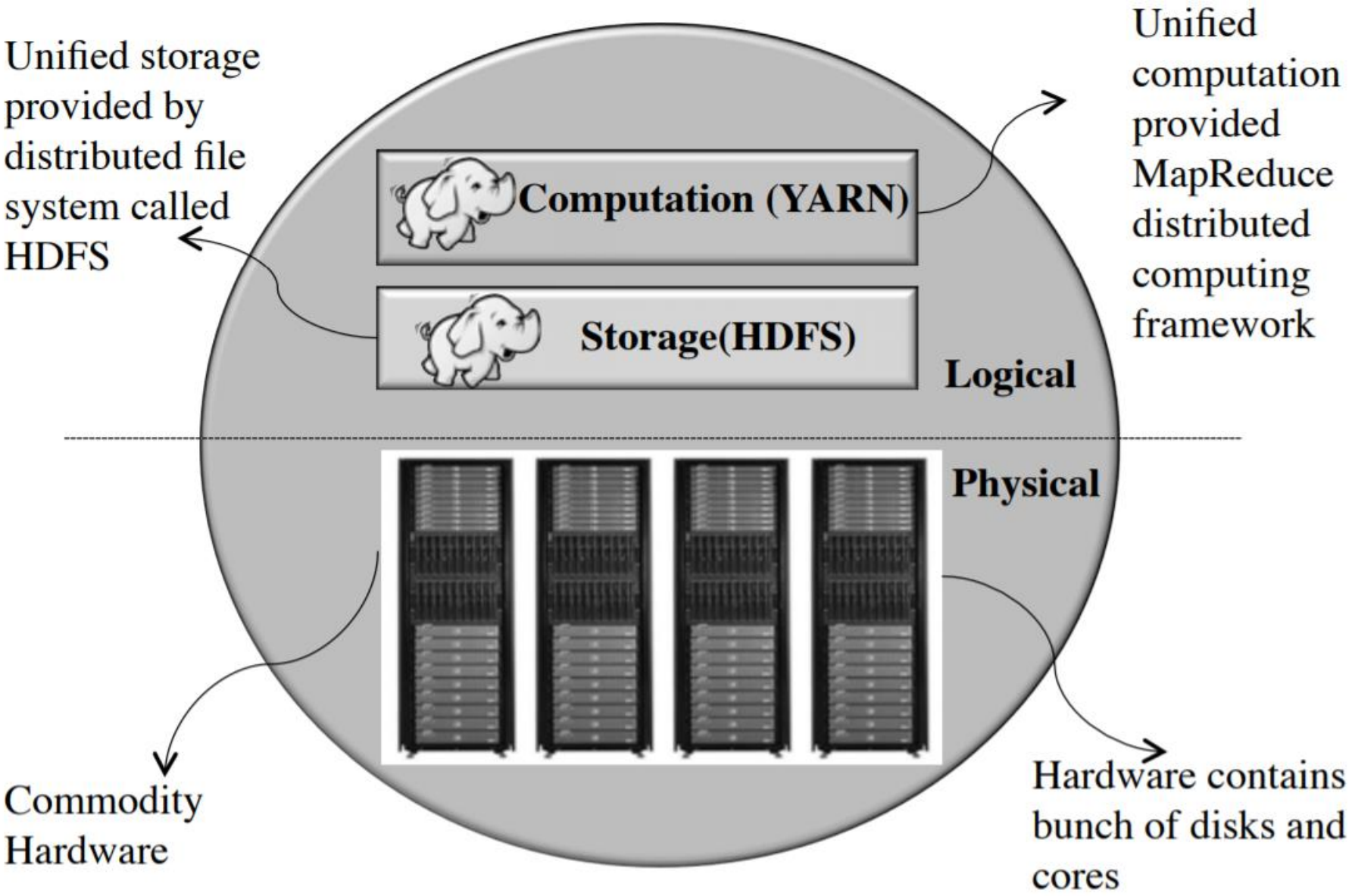
Big Data to Hadoop

- Characteristics - Volume, Variety and Velocity
- Batch
  - Hadoop with Map Reduce
  - GFS -> HDFS
  - Map Reduce -> Hadoop Map Reduce
- Operational but not Transactional
  - NoSQL
  - Google Big Table -> HBase

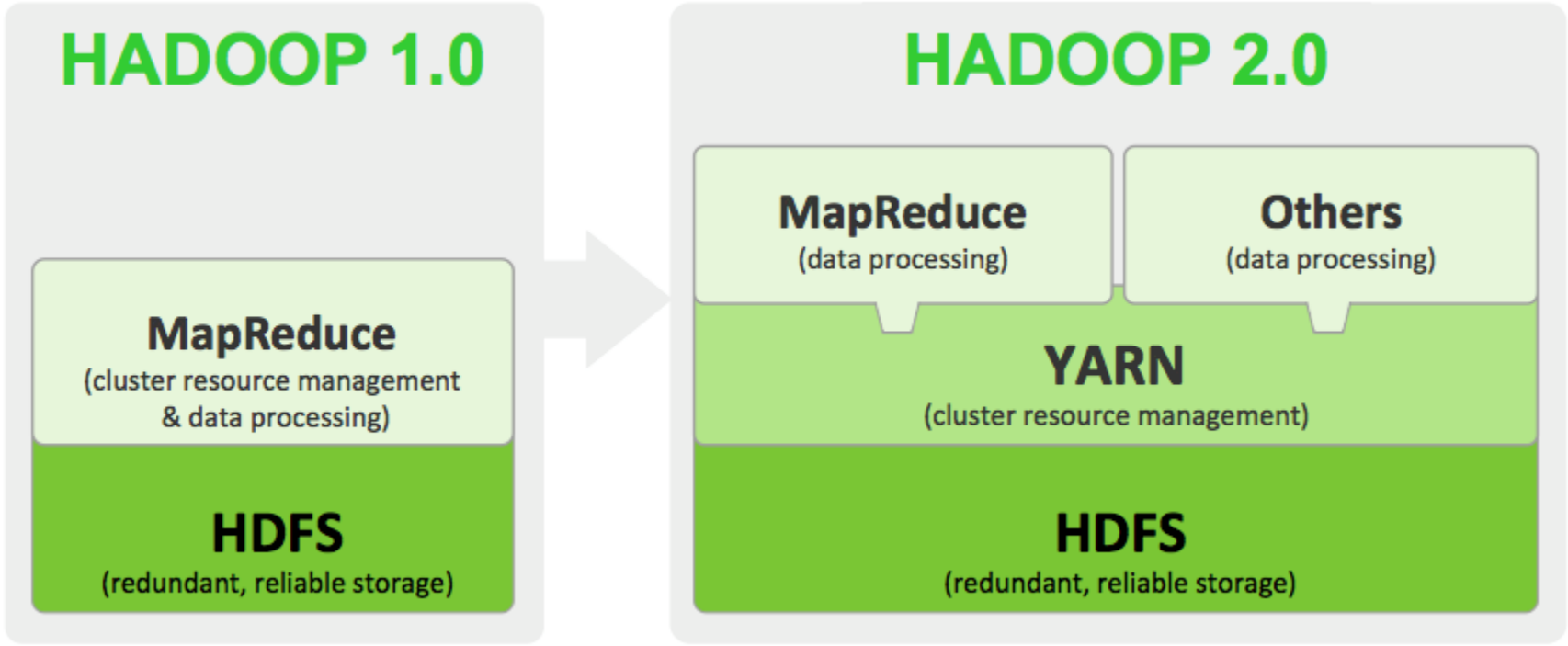
Hadoop core components

- **Hadoop Distributed File System (HDFS) –** It is the storage layer of Hadoop.
- **Map-Reduce –** It is the data processing layer of Hadoop.
- **YARN –** It is the resource management layer of Hadoop.

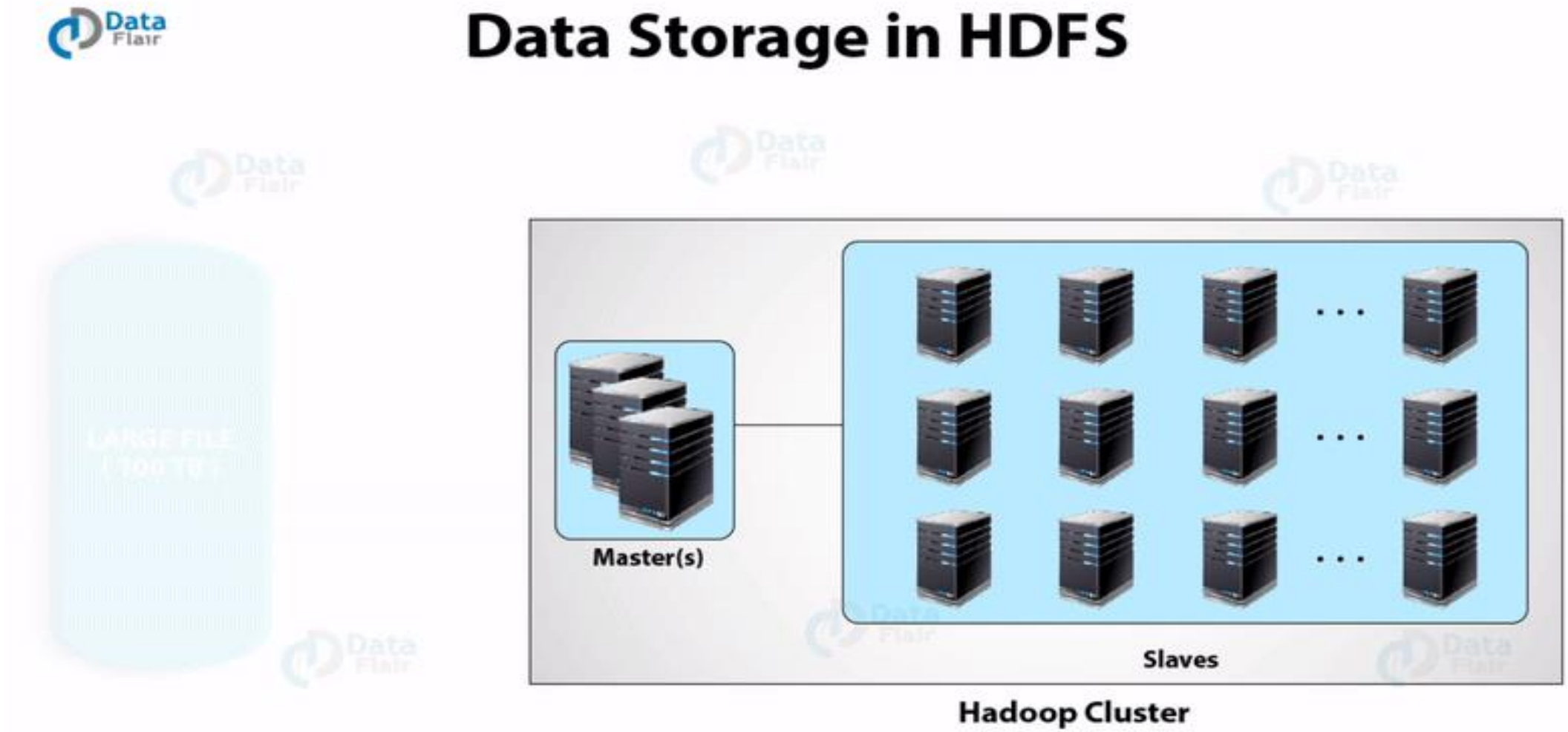
Hadoop – Big Picture



Hadoop Versions

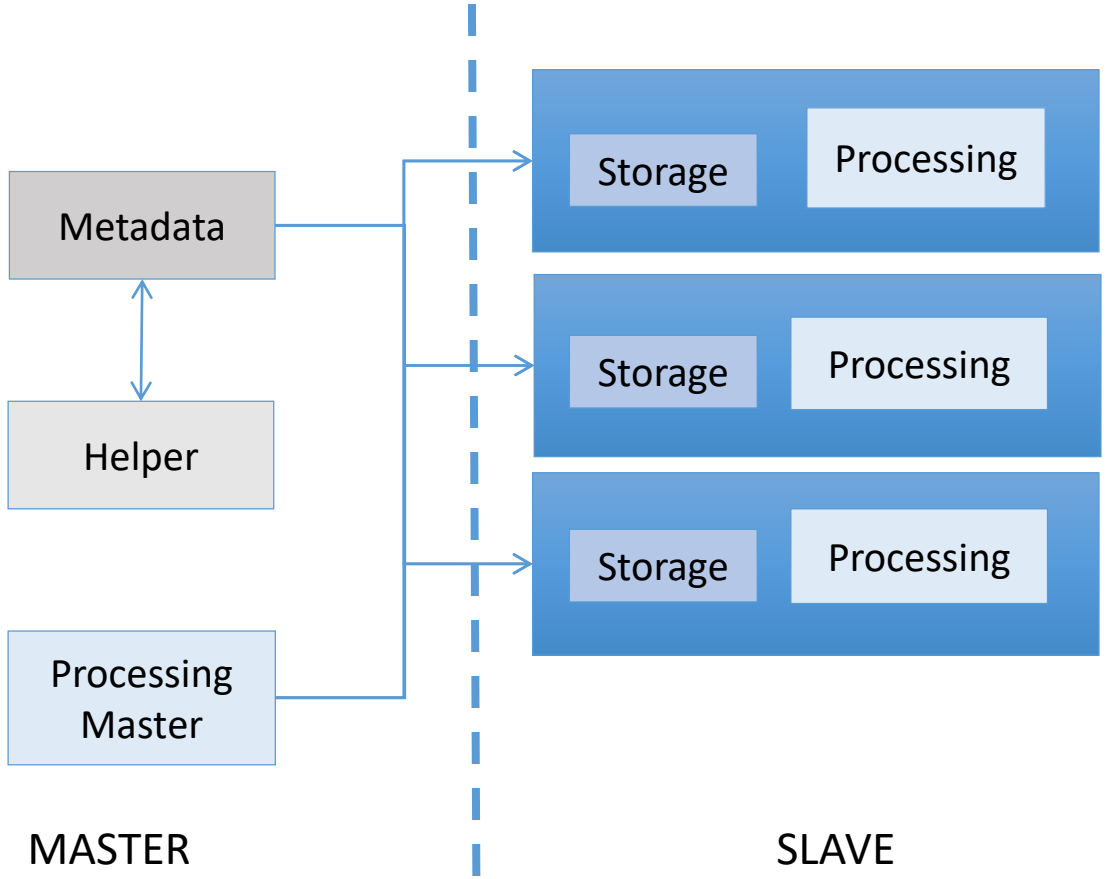


Data Storage in HDFS



Hadoop Architecture

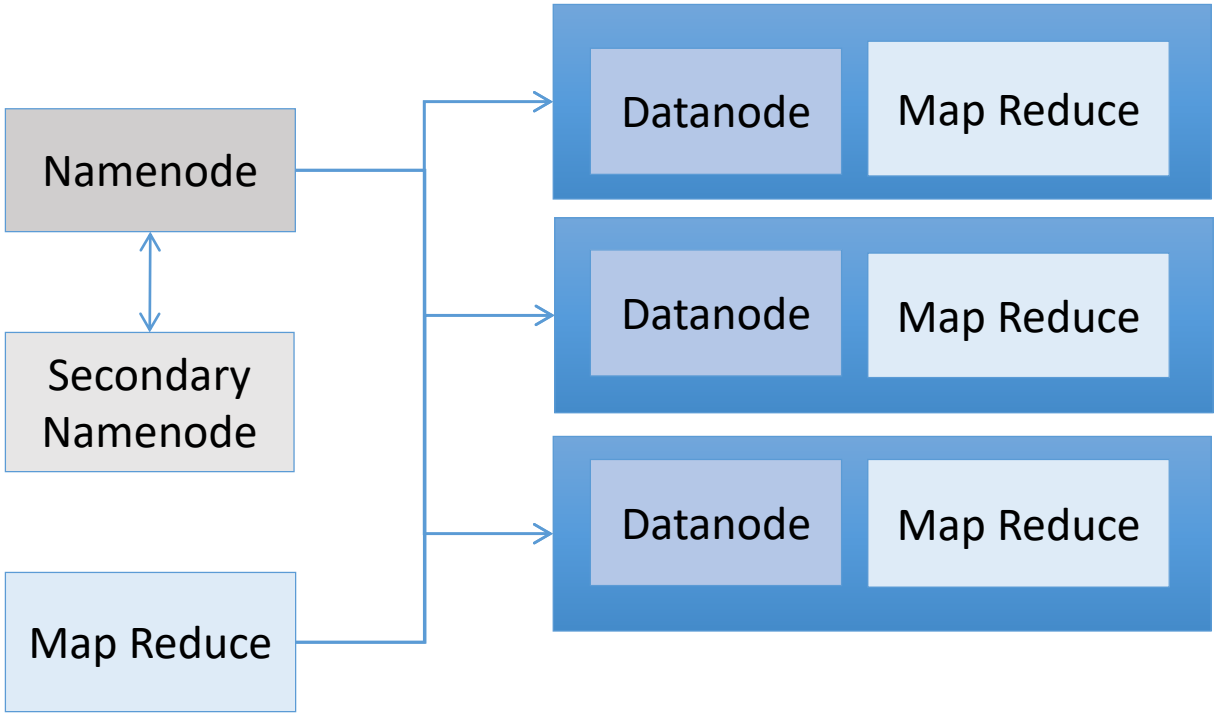
Master is a **high-end machine** where as slaves are inexpensive computers. The Big Data files get divided into the **number of blocks**. Hadoop stores these blocks in a distributed fashion on the cluster of slave nodes. On the master, we have **metadata stored**.



HDFS Architecture

- HDFS has **two daemons** running for it. They are :
- **NameNode :** NameNode performs following functions –
  - NameNode Daemon runs on the master machine.
  - It is responsible for maintaining, monitoring and managing DataNodes.
  - It records the metadata of the files like the location of blocks, file size, permission, hierarchy etc.
  - Namenode captures all the changes to the metadata like deletion, creation and renaming of the file in edit logs.
  - It regularly receives heartbeat and block reports from the DataNodes.
- **DataNode:** The various functions of DataNode are as follows –
  - DataNode runs on the slave machine.
  - It stores the actual business data.
  - It serves the read-write request from the user.
  - DataNode does the ground work of creating, replicating and deleting the blocks on the command of NameNode.
  - After every 3 seconds, by default, it sends heartbeat to NameNode reporting the health of HDFS.

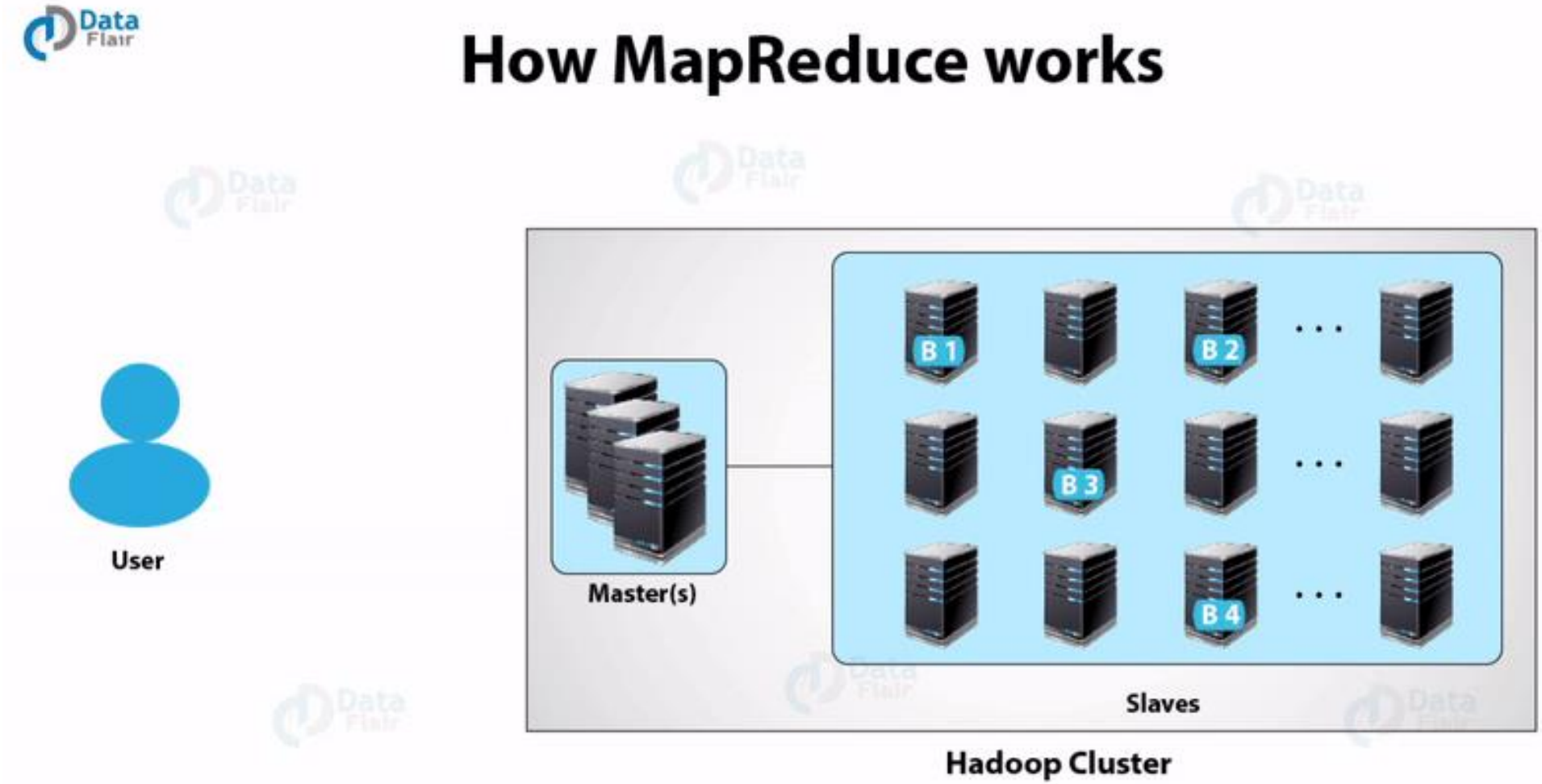
HDFS Architecture



Hadoop Processing

- It is the data processing layer of Hadoop. It processes data in two phases:
- **Map Phase-** This phase applies business logic to the data. The input data gets converted into key-value pairs.
- **Reduce Phase-** The Reduce phase takes as input the output of Map Phase. It applies aggregation based on the key of the key-value pairs.
- Versions:
  - MRv1/Classic
  - MRv2/YARN

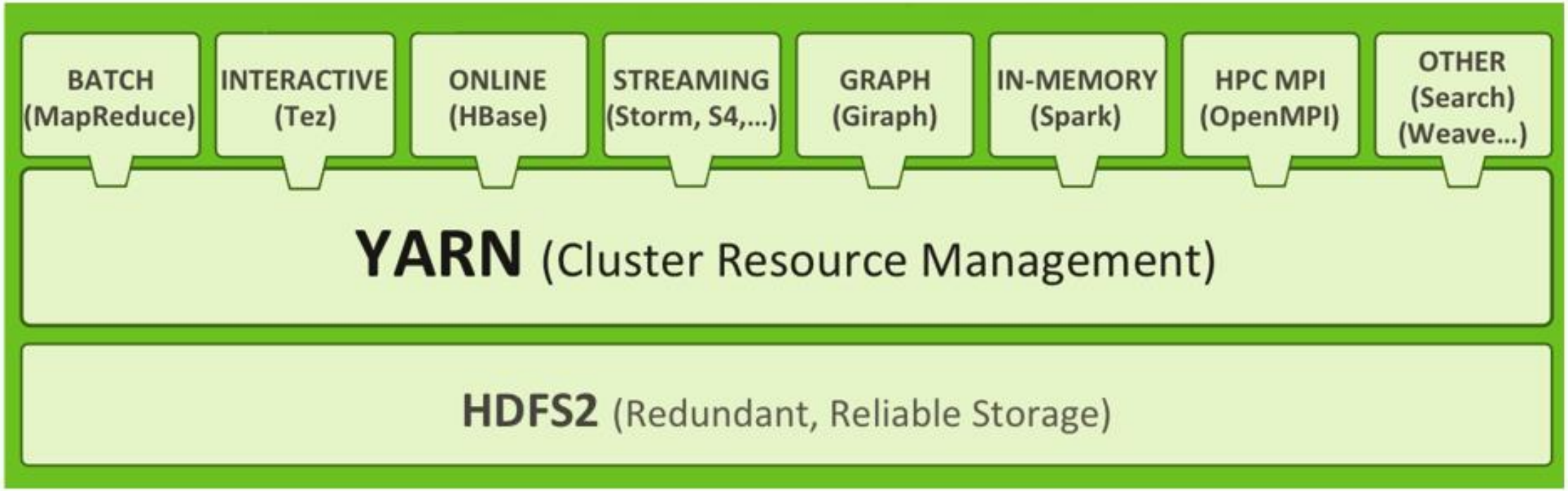
How MapReduce Works?



Map-Reduce works in the following way

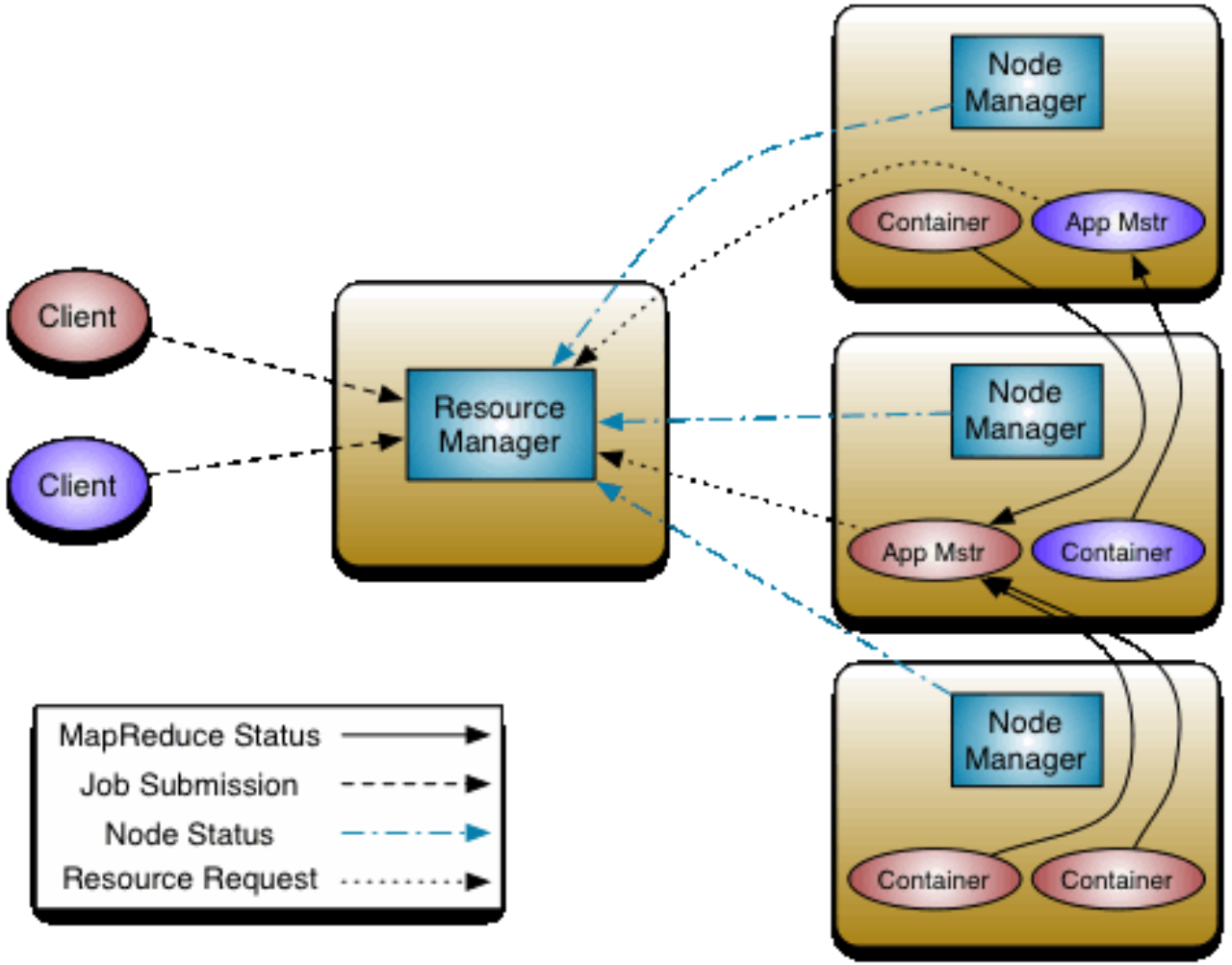
- The client specifies the file for input to the Map function. It splits it into tuples
- Map function defines key and value from the input file. The output of the map function is this key-value pair.
- MapReduce framework sorts the key-value pair from map function.
- The framework merges the tuples having the same key together.
- The reducers get these merged key-value pairs as input.
- Reducer applies aggregate functions on key-value pair.
- The output from the reducer gets written to HDFS.

YARN Architecture



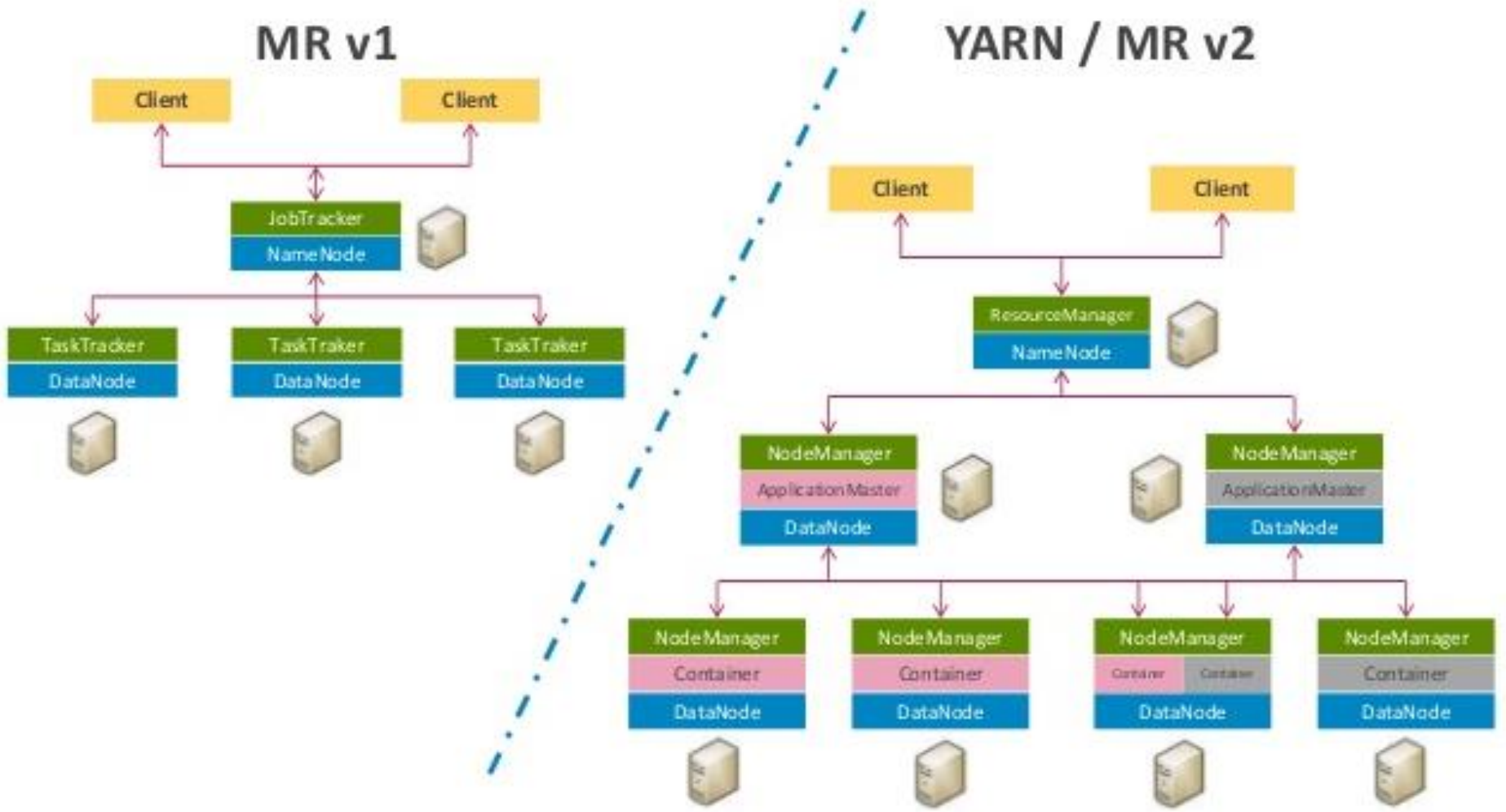
Apache Hadoop coupled with various frameworks to support data processing for all types of data.

YARN Architecture MRv2

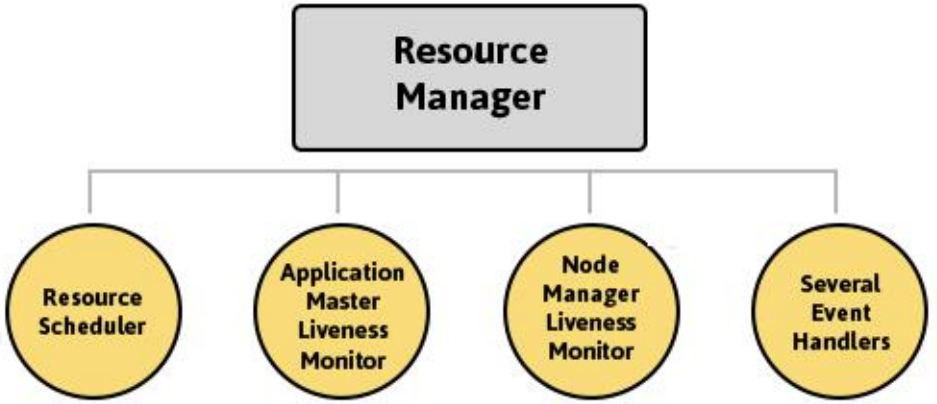


MR vs YARN Architecture

- YARN
  - Yet Another Resource Negotiator
- MR
  - MapReduce

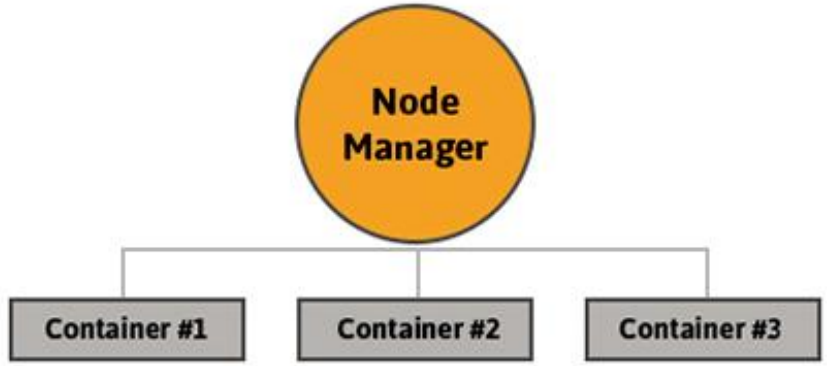


YARN Resource Manager



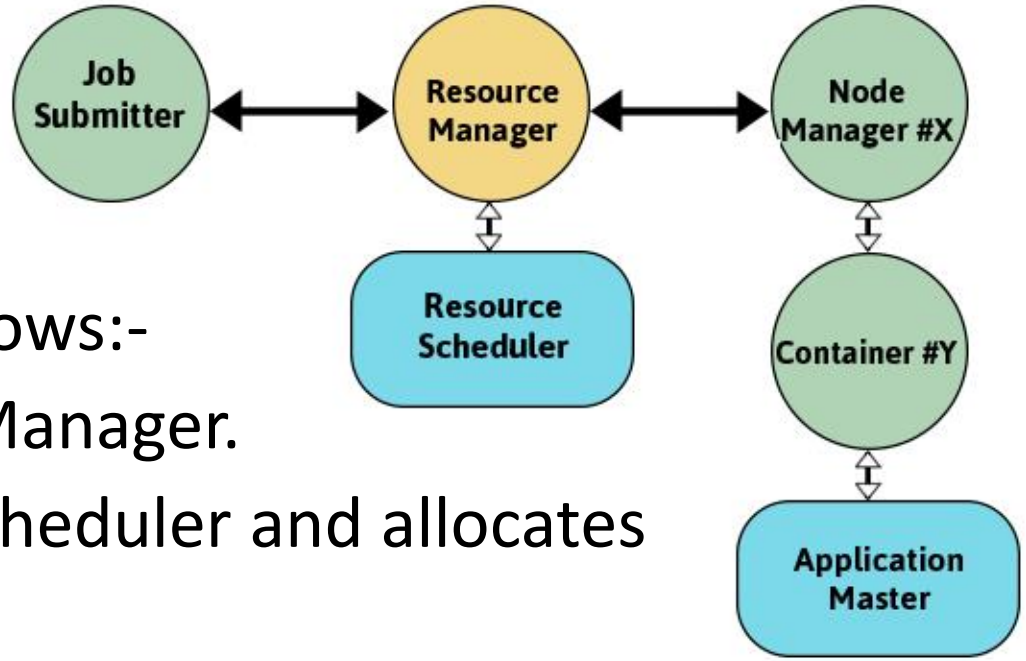
- Resource Manager runs on the master node.
- It knows where the location of slaves (Rack Awareness).
- It is aware about how much resources each slave have.
- Resource Scheduler is one of the important service run by the Resource Manager.
- Resource Scheduler decides how the resources get assigned to various tasks.
- Application Manager is one more service run by Resource Manager.
- Application Manager negotiates the first container for an application.
- Resource Manager keeps track of the heart beats from the Node Manager.

YARN Node Manager



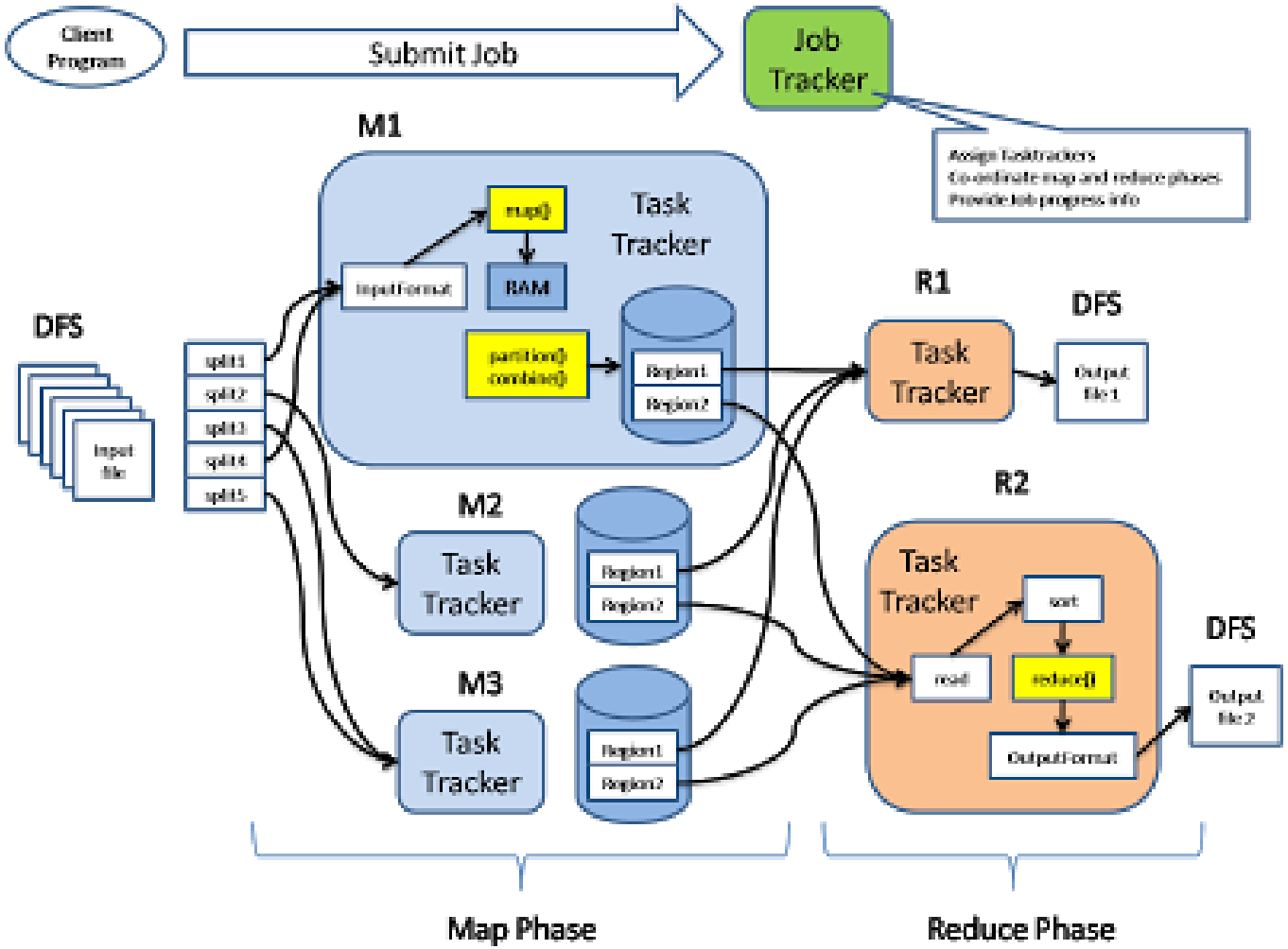
- It runs on slave machines.
- It manages containers. Containers are nothing but a fraction of Node Manager’s resource capacity
- Node manager monitors resource utilization of each container.
- It sends heartbeat to Resource Manager.

YARN Job Submitter

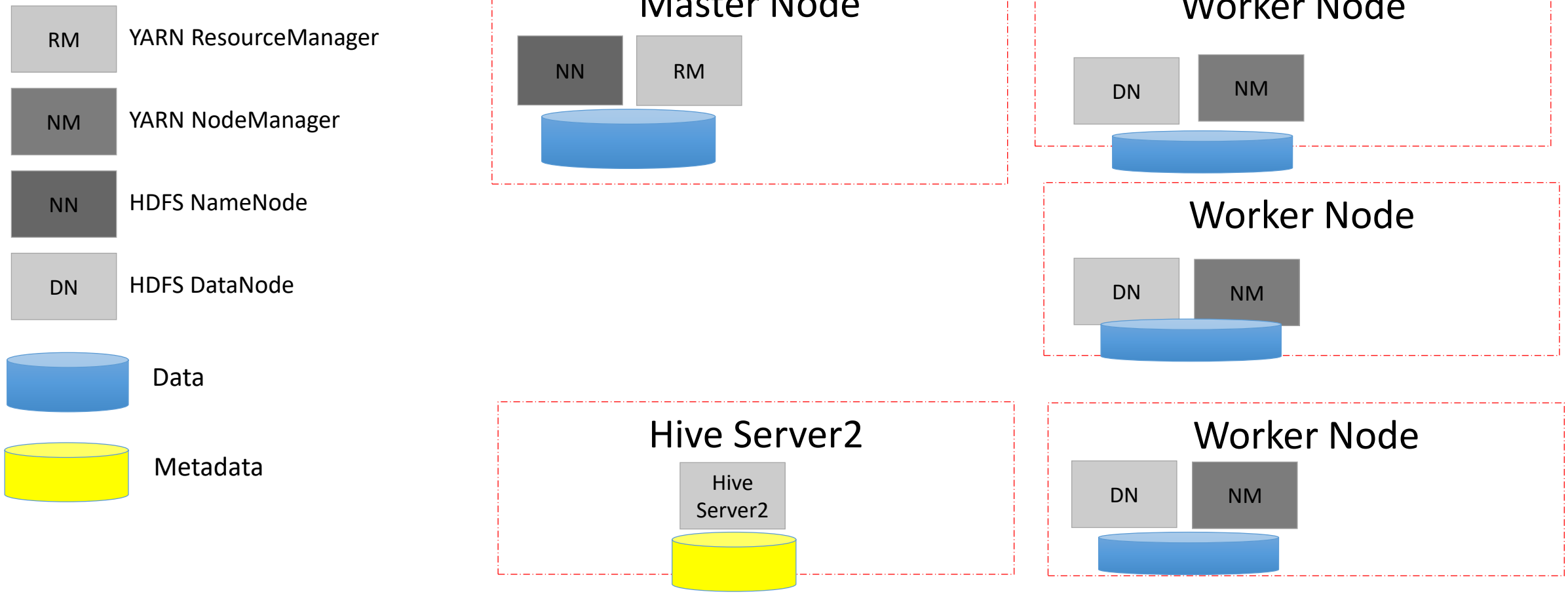


- The application startup process is as follows:-
- The client submits the job to Resource Manager.
- Resource Manager contacts Resource Scheduler and allocates container.
- Now Resource Manager contacts the relevant Node Manager to launch the container.
- Container runs Application Master.
- The basic idea of YARN was to split the task of resource management and job scheduling. It has one global Resource Manager and per-application Application Master. An application can be either one job or DAG of jobs.

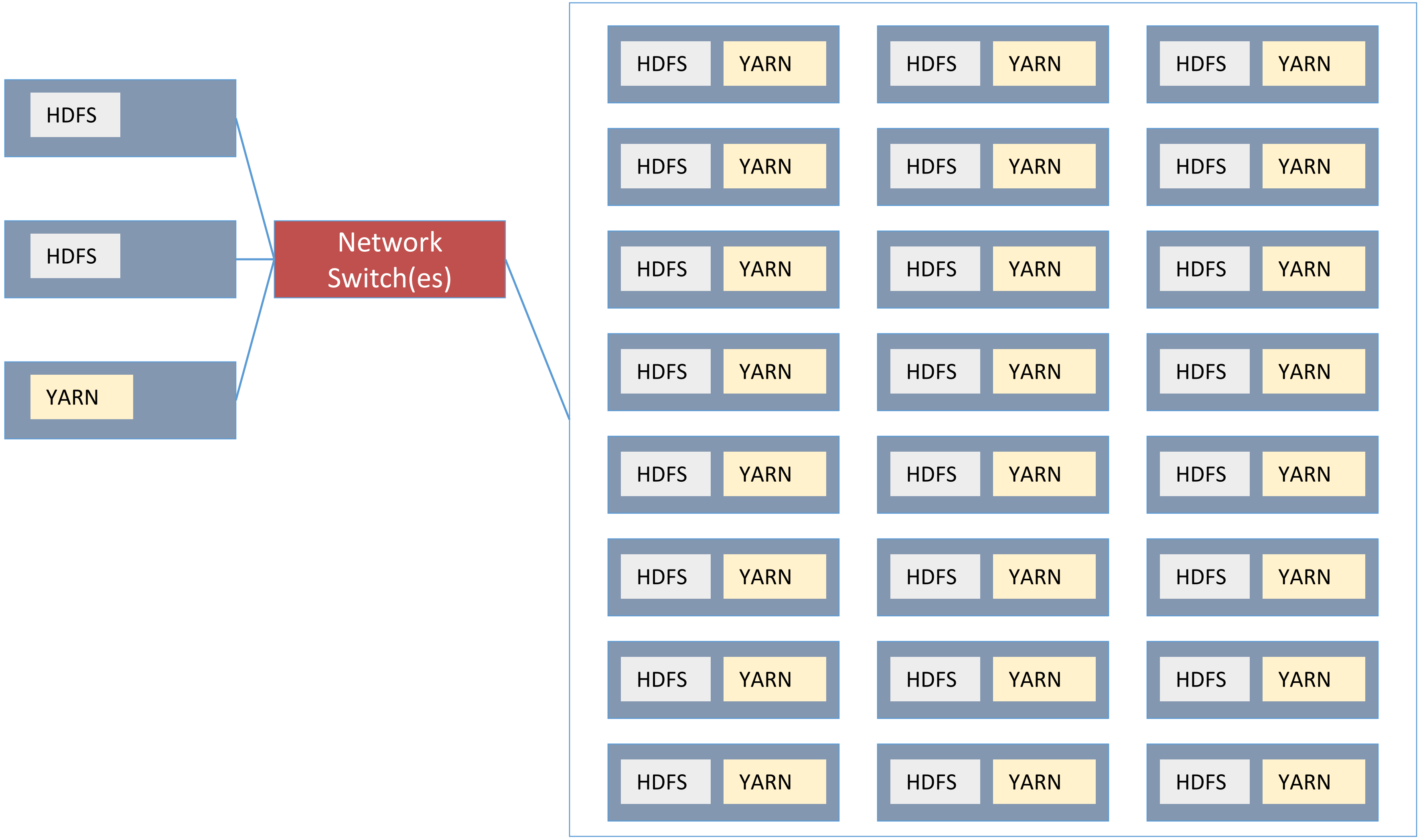
YARN Job Execution



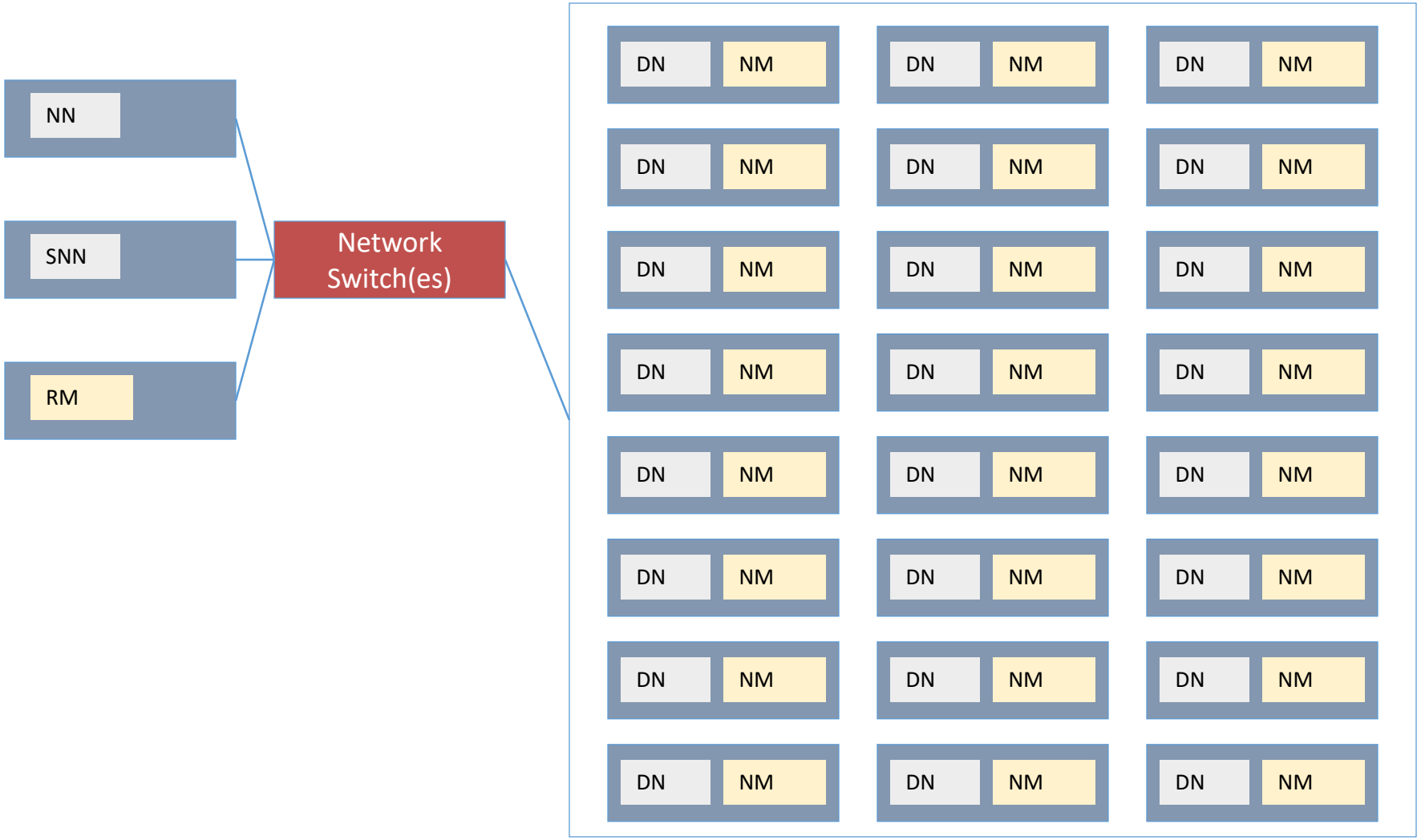
Hadoop itself is clustered....



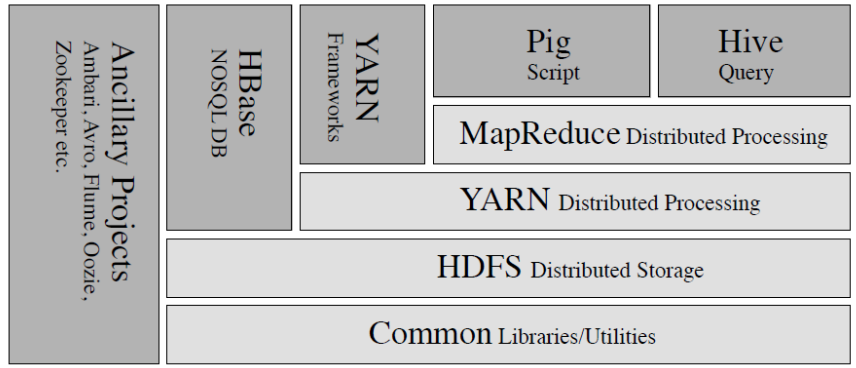
Typical Hadoop Cluster

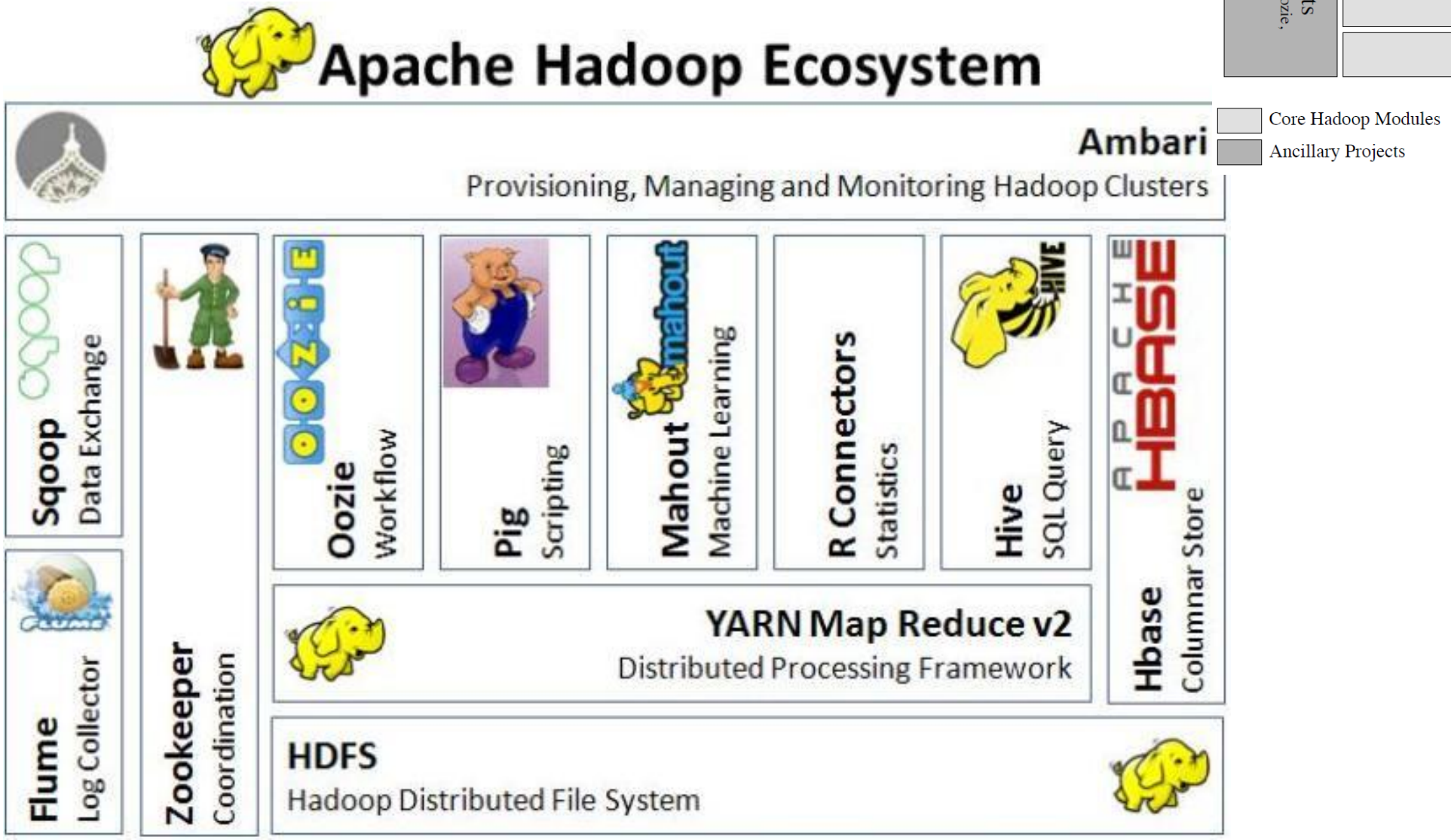


Typical Hadoop Cluster



Hadoop Technology Stack



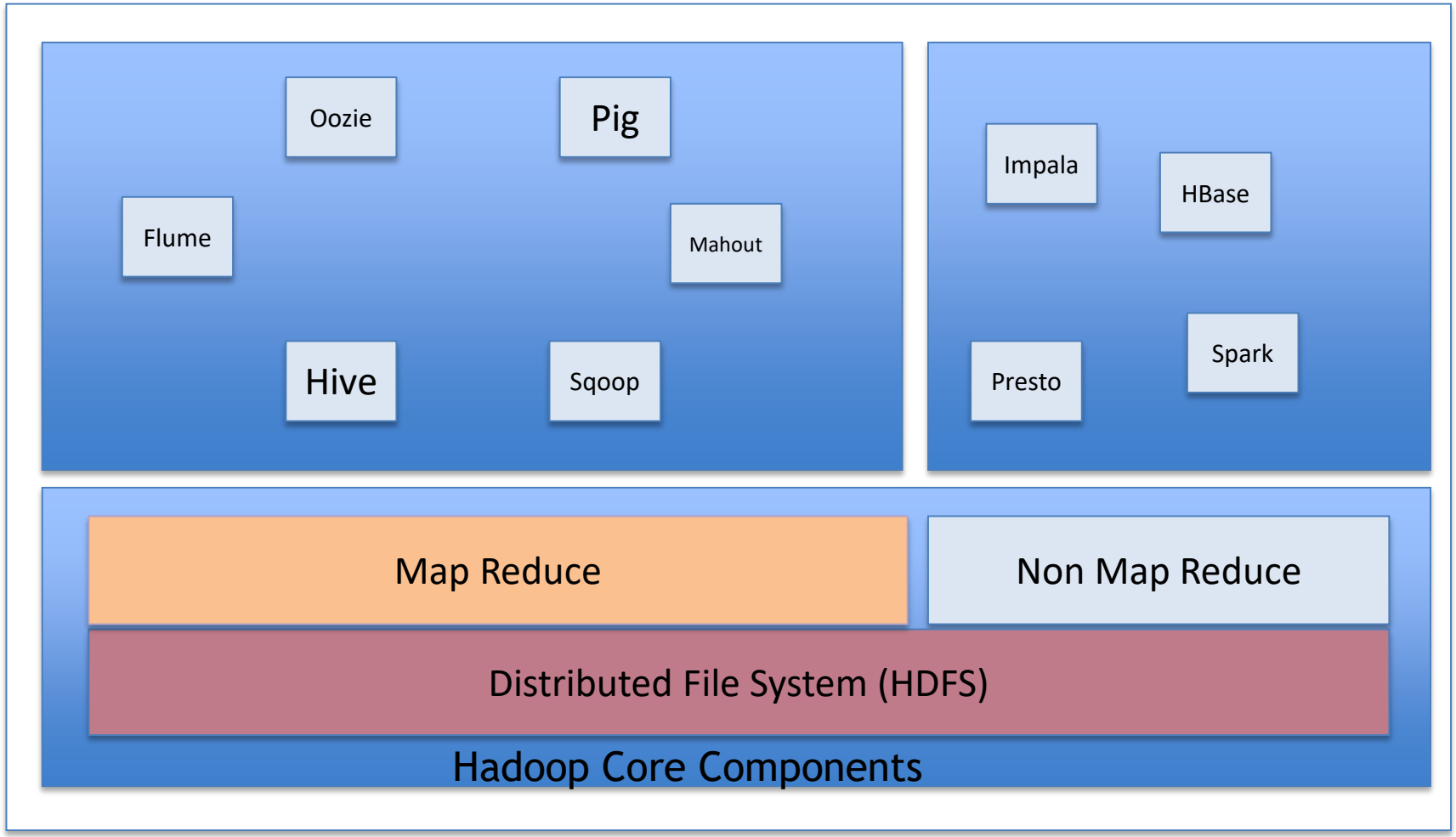


Application vs Service vs Instance

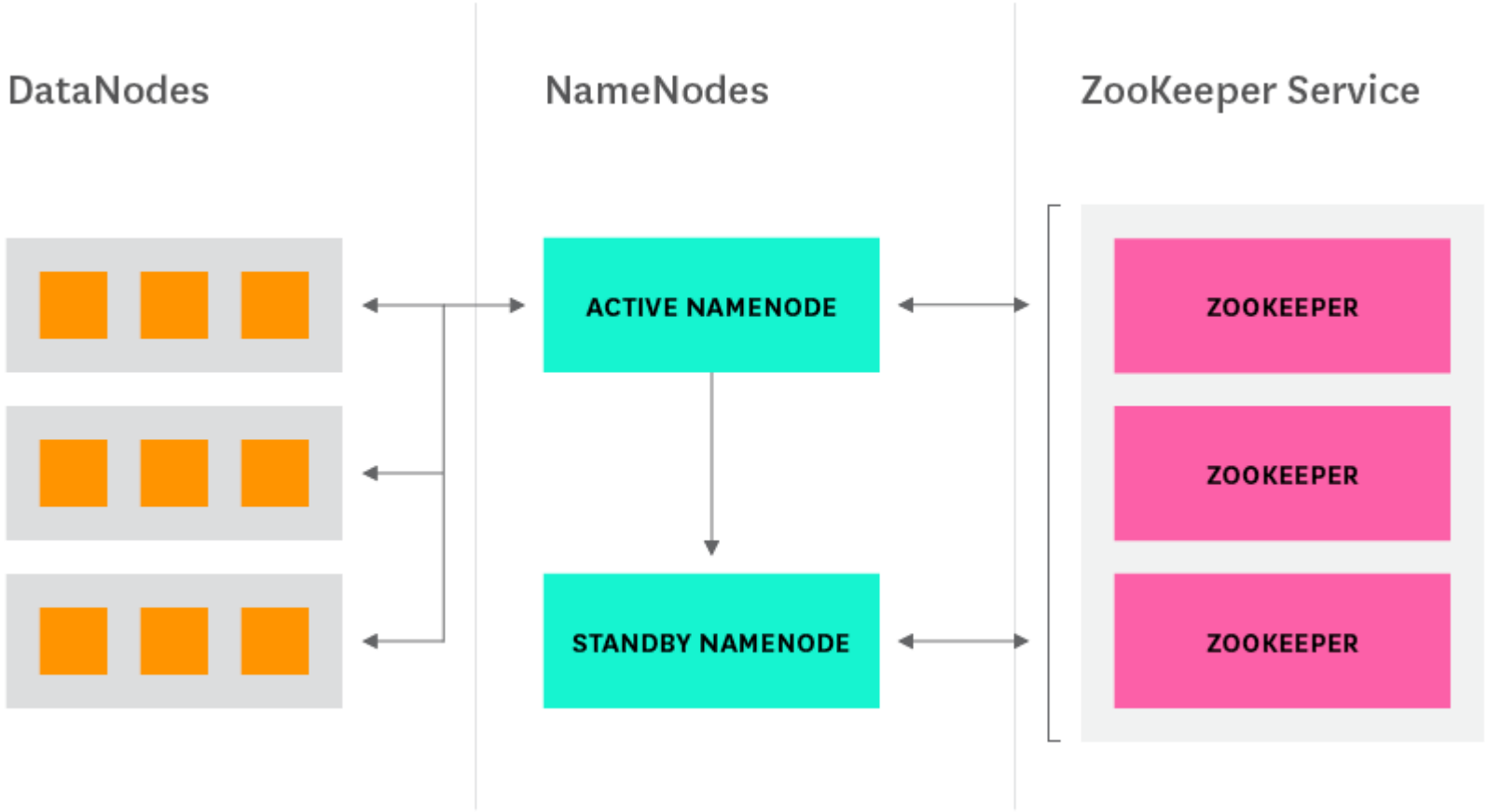
- For example, “Hadoop” is an application
- “Collection of Services” - NodeManager, DataNode, ResourceManager are application services
- The ResourceManager Service running on node host-1.example.com is an application service instance

Hadoop EcoSystem

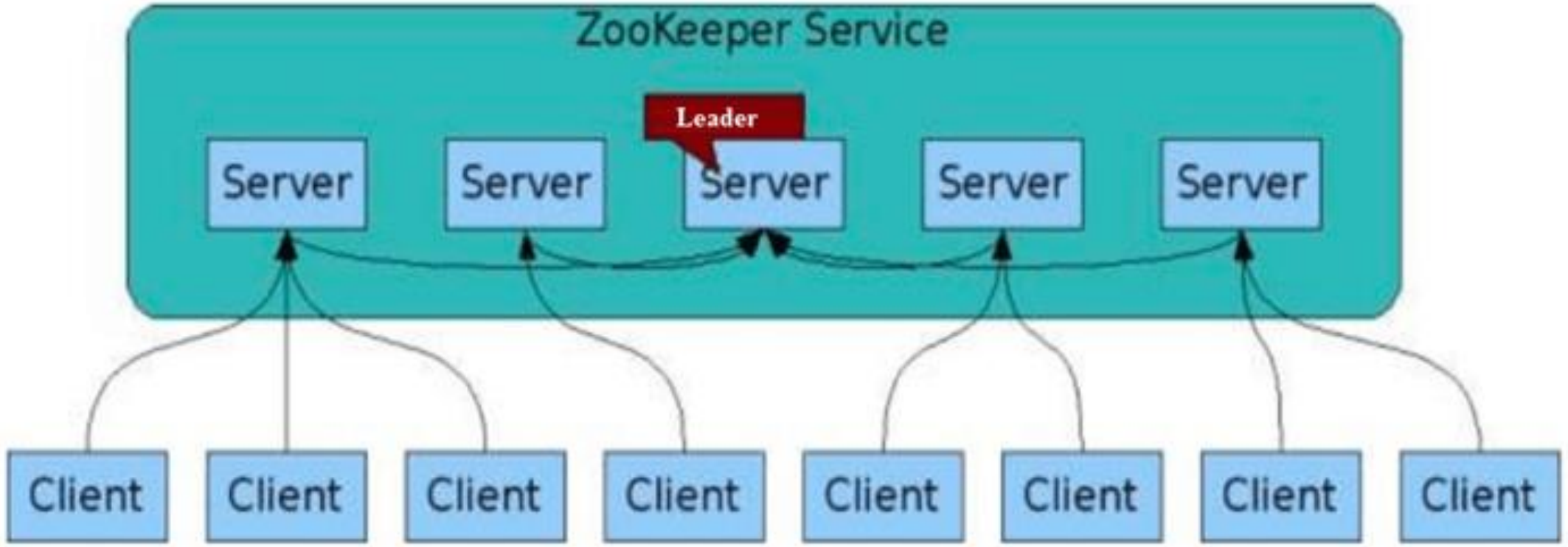
Hadoop Components



Hadoop High Availability



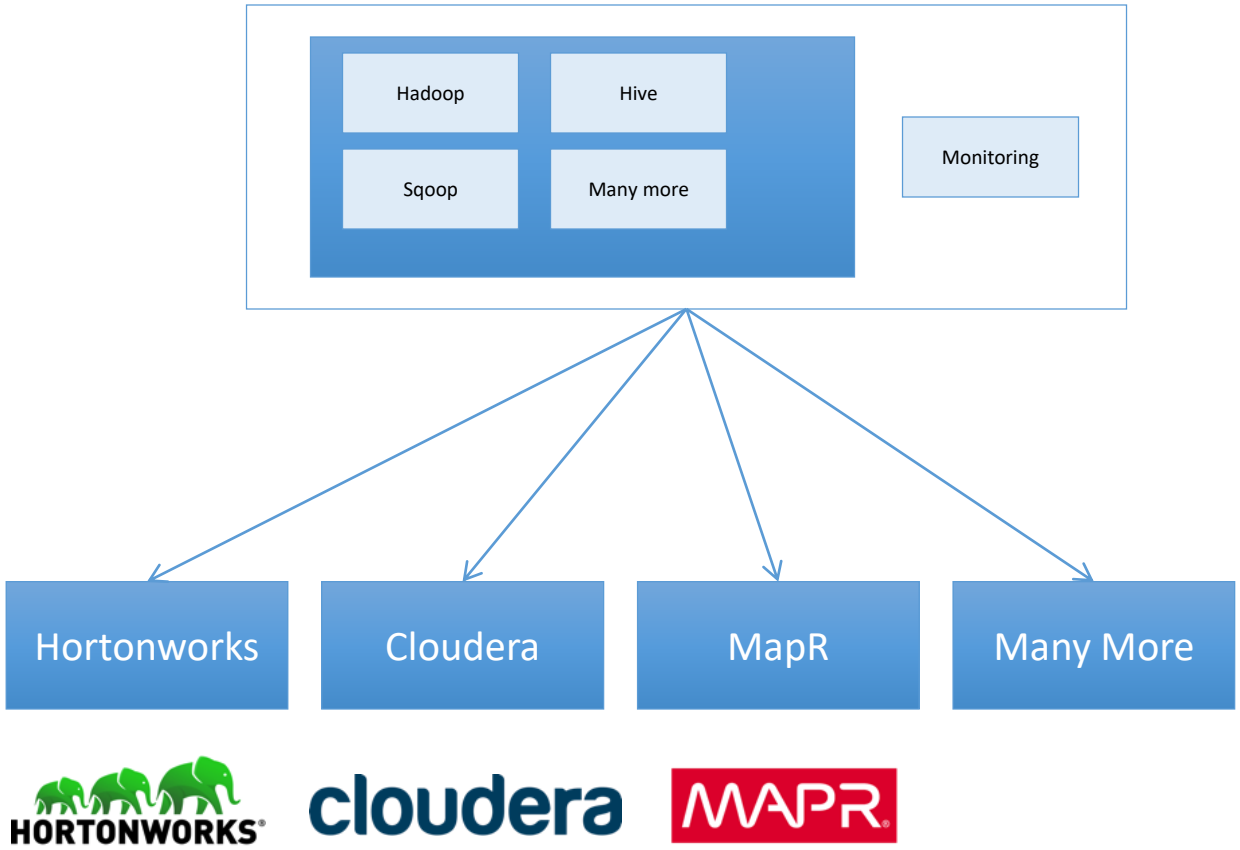
Hadoop Zookeeper Architecture



Concept of Leader / Followers

Hadoop Distributions

- **Apache** – Vanilla flavor, as the actual code is residing in Apache repositories.
- **Hortonworks** – Popular distribution in the industry.
- **Cloudera** – It is the most popular in the industry.
- **MapR** – It has rewritten HDFS and its HDFS is faster as compared to others.
- **IBM** – Proprietary distribution is known as Big Insights.



Hadoop Distributions

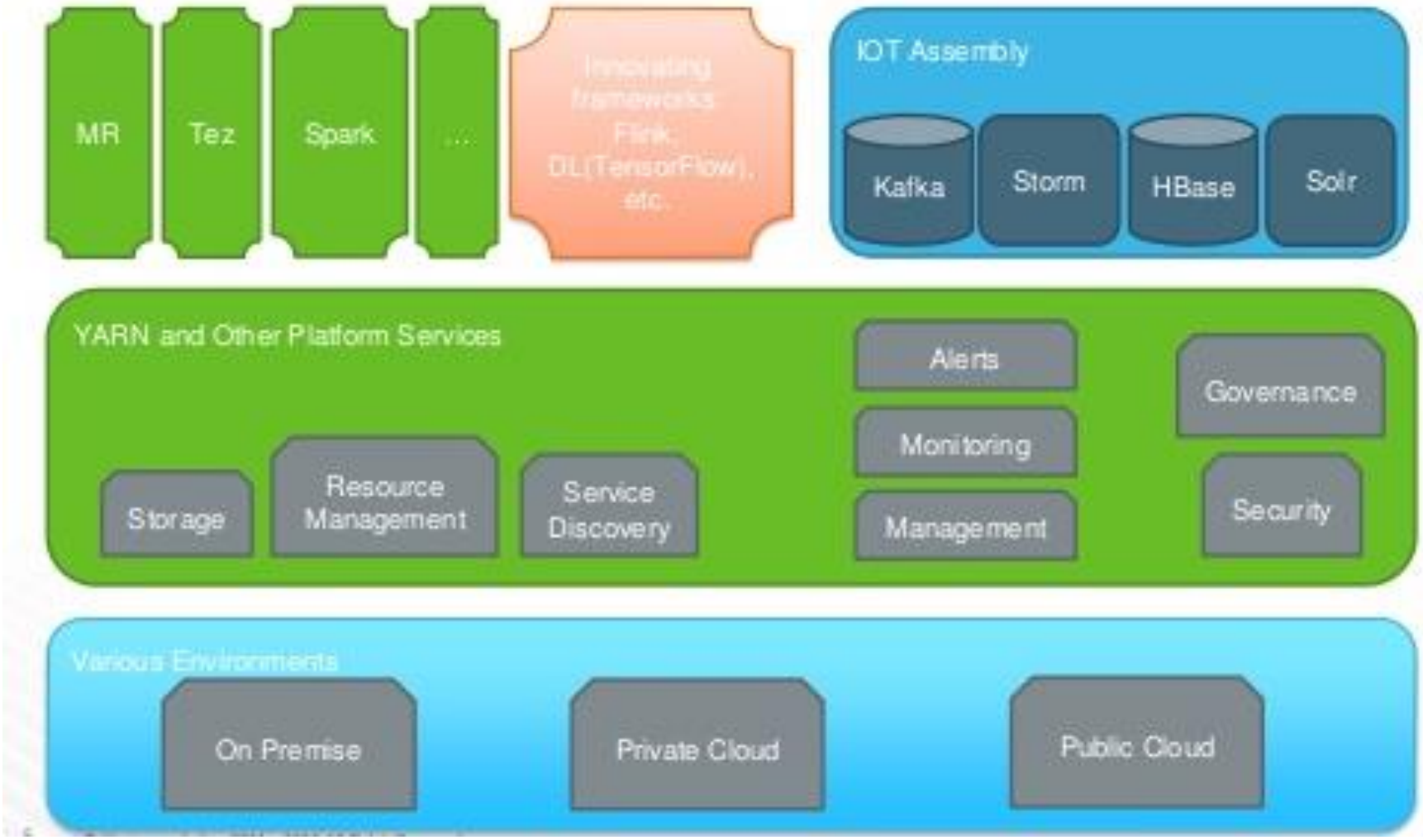
- Hadoop and eco system tools are Apache open source projects
- Cloudera, Hortonworks and other leading Hadoop based technology companies commit to these open source projects
- They provide training, support and services.
- Cloudera have proprietary monitoring tool developed for large hadoop clusters. It is free up to 50 nodes after which license fee needs to be paid.
- Hortonworks uses Ambari which is a open source monitoring tool. No license fee

Complete list of Hadoop Services?

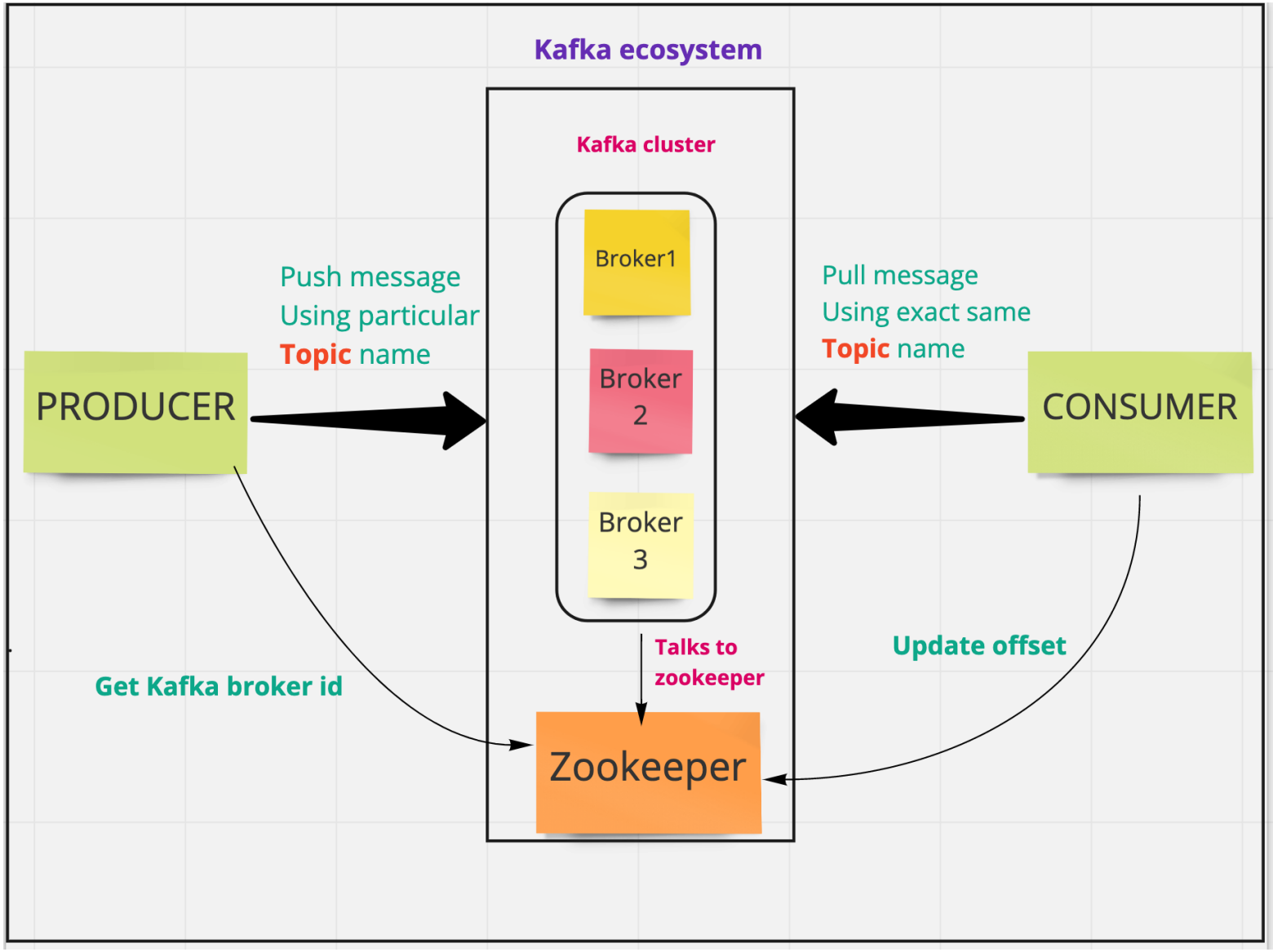
RM	YARN ResourceManager
NM	YARN NodeManager
NN	HDFS NameNode
DN	YARN DataNode
JHS	Job History Server
HFS	HttpFS Service
JN	Journal Node
ZK	ZooKeeper
HM	Hbase Master
HRS	Hbase Region Server
SHS	Spark History Server
Hue	Hue
OZ	Oozie
CM	Cloudera Manager
DB	RDBMS
GW	Gateway
FA	Flume Agent
ISS	Impala State Store
ICS	Impala Catalog Server
ID	Impala Daemon
SS	Solr Server
HS	Hive Server
HSS	Hive Metastore Service
	…

ACK! Seemingly no end to the Big Data services.

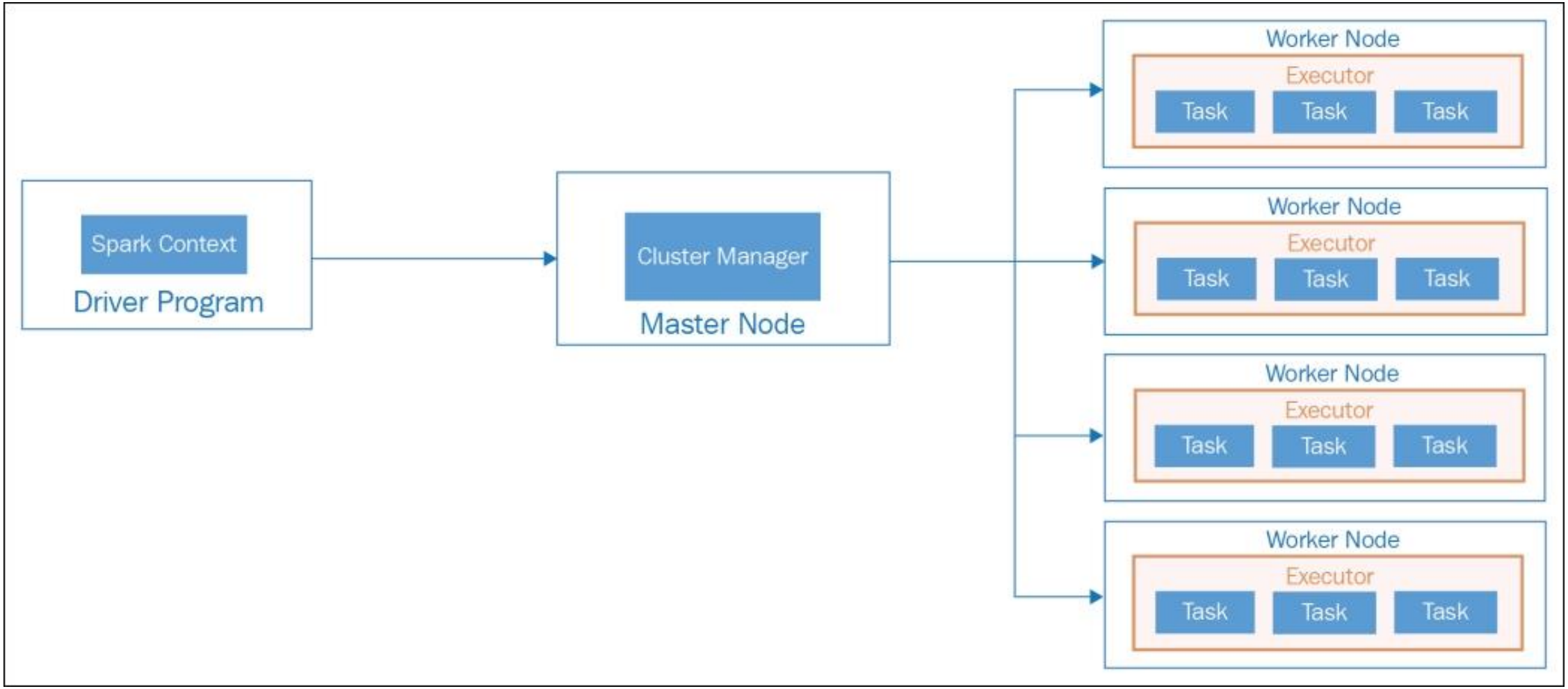
Hadoop 3.0 Architecture



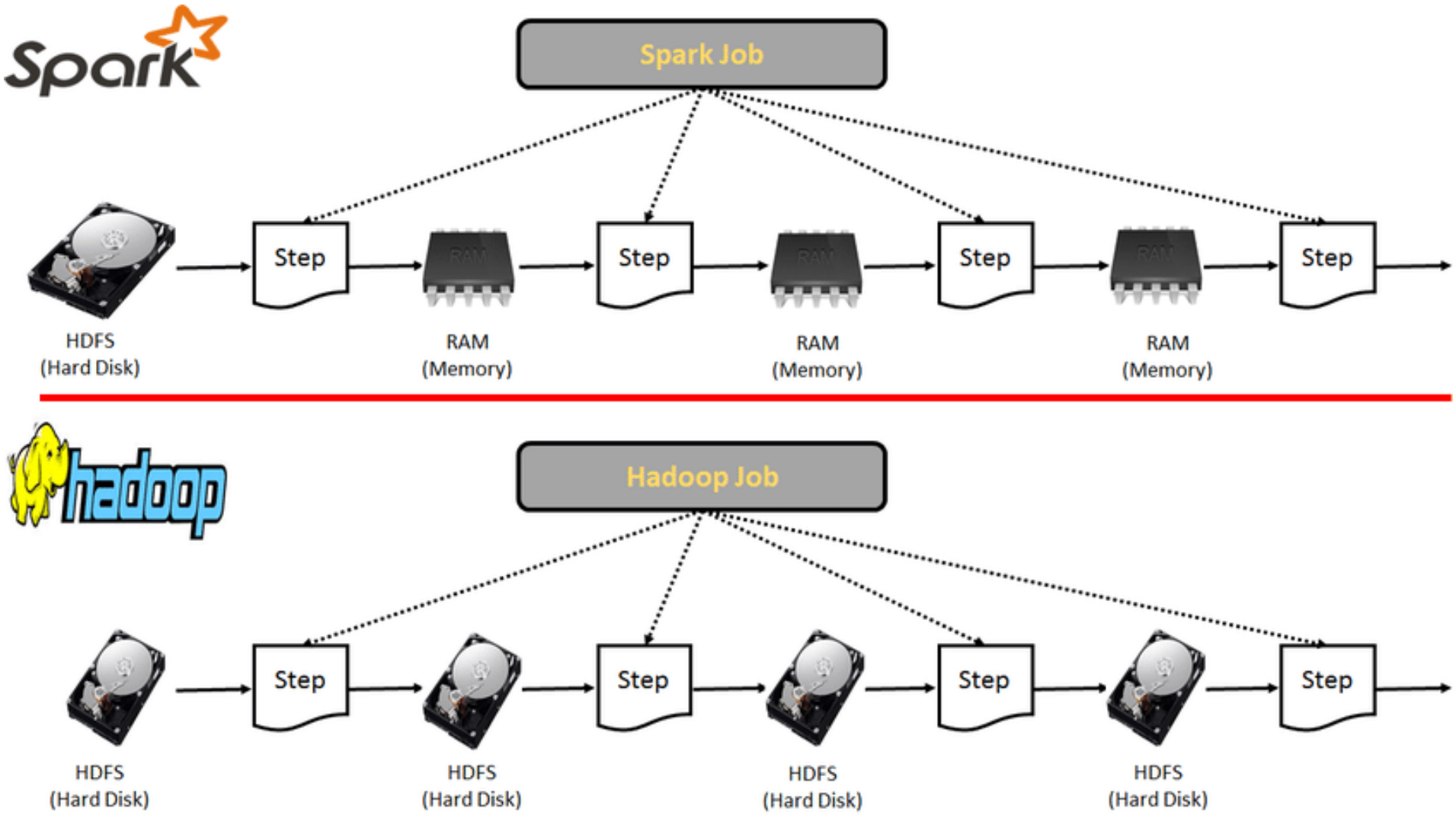
Kafka Architecture for Streaming



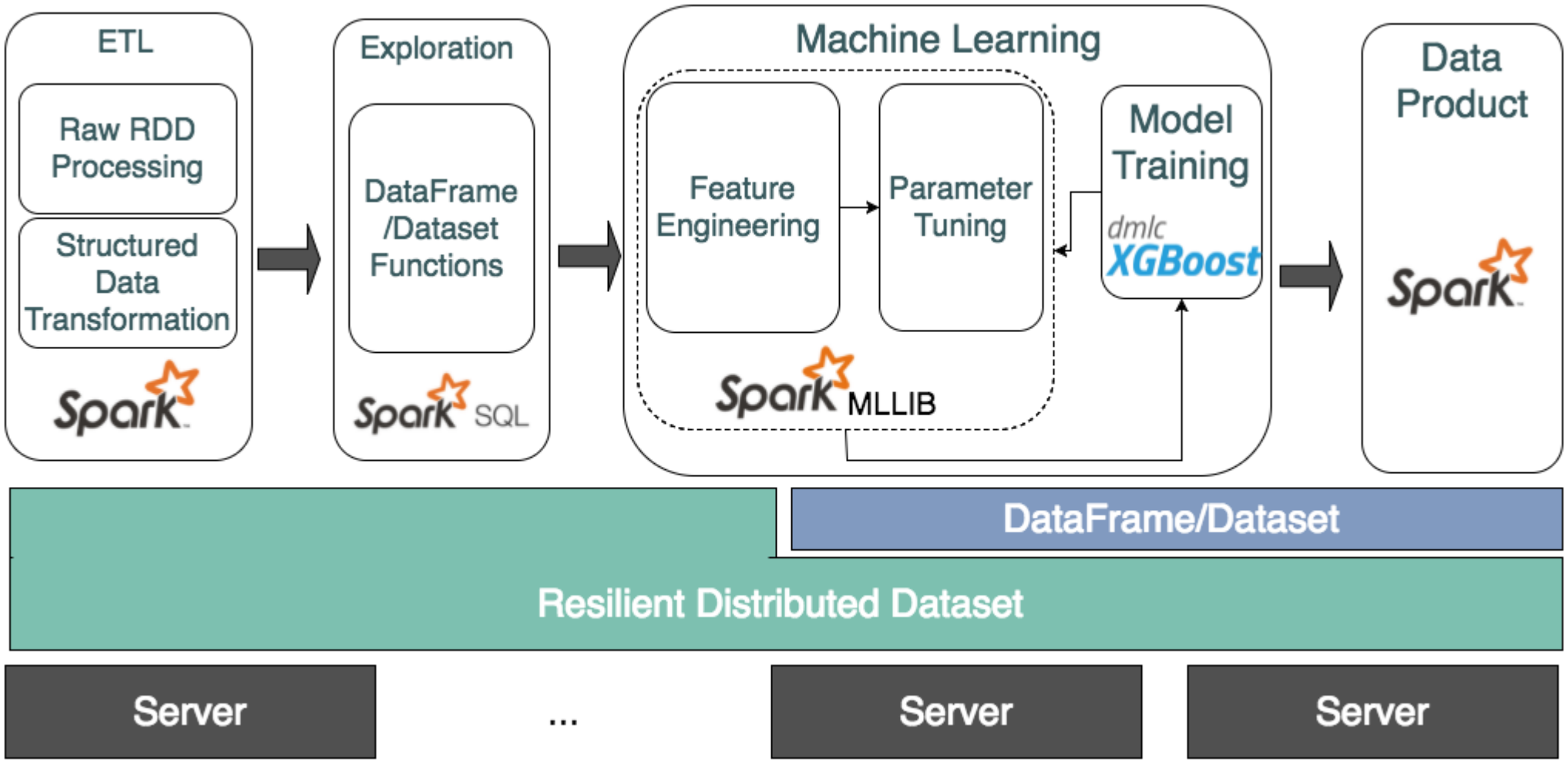
Spark Architecture



Spark vs Hadoop MapReduce



Data analytics with Spark



PySpark implementation (Keras)

```
def predict(iterator):
 model = InceptionV3(weights=None)
 model.load_weights(FLAGS.weights_file)
 return [(x[0], run_inference_on_image(model, x[1])) for x in iterator]


def main():
 sc = SparkContext(conf=conf)
 hbase_io = common.HbaseIO(FLAGS)
 out_format = common.OutputFormatter(FLAGS, MODEL_NAME)

 hbase_images = hbase_io.load_from_hbase(sc)
 classified_images = hbase_images.mapPartitions(predict) \
   .map(out_format.imagenet_format)
 classified_images.foreachPartition(hbase_io.put_to_hbase)
```

PySpark implementation (Keras)

- The Python environment with tensorflow is distributed to the executors at runtime, it is not preinstalled on the nodes.
- The individual models only need to implement the following functions:
  - prepare
  - predict
  - output_format
- Conceptually very close to scikit-learn or Spark ML Pipelines approach
- Deep Learning Pipelines can be a way to streamline the implementation

https://databricks.com/blog/2017/06/06/databricks-vision-simplify-large-scale-deep-learning.html

Spark implementation (dl4j / Scala)

```
def predict(pairs: Iterator[(String, (INDArray, Int, Int))]) = {
 val model = ModelSerializer.restoreComputationGraph(modelLoc)
 pairs.map{ case (name, image) =>
        (name, run_inference_on_image(model, image)
 }
}

def main(args: Array[String]) = {
 val sc = SparkContext(conf=conf)
 val hbase_io = common.HbaseIO(args)
 val out_format = common.OutputFormatter(args)

 val hbase_images = hbase_io.load_from_hbase(sc)
 val classified_images = hbase_images.mapPartitions(predict) \
   .map(out_format.imagenet_format)
 val classified_images.foreachPartition(hbase_io.put_to_hbase)
}
```

Image Clustering using KMeans

- Extraction of image features
- Conversion of specific data formats into:

```
org.apache.spark.mllib.linalg.{Vector, Vectors}
```

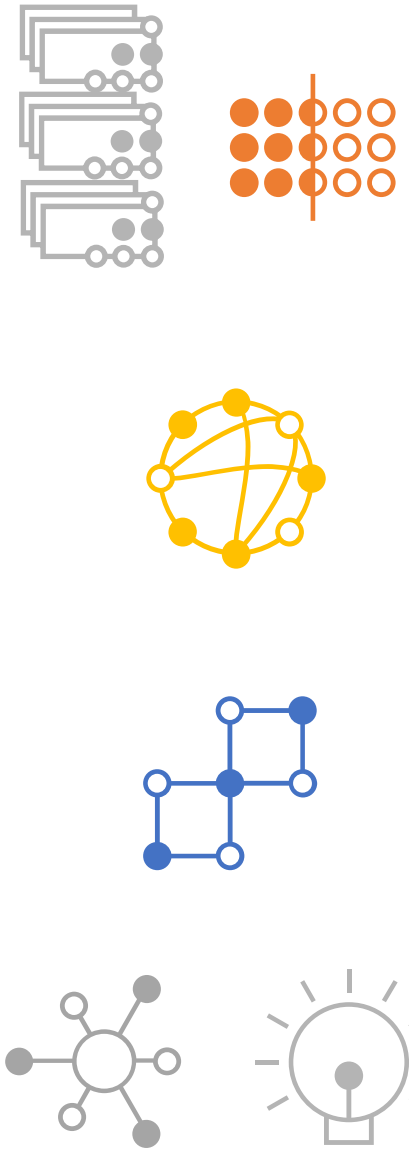
- Features are persisted in this reusable format as a Parquet file

- From here we go ... apply SparkML code to a part of the compound dataset.

- Finally we feed the new labels back into the compound dataset
  ➔ e.g., for comparison of different clustering models with known clusters

./runSpark2.sh

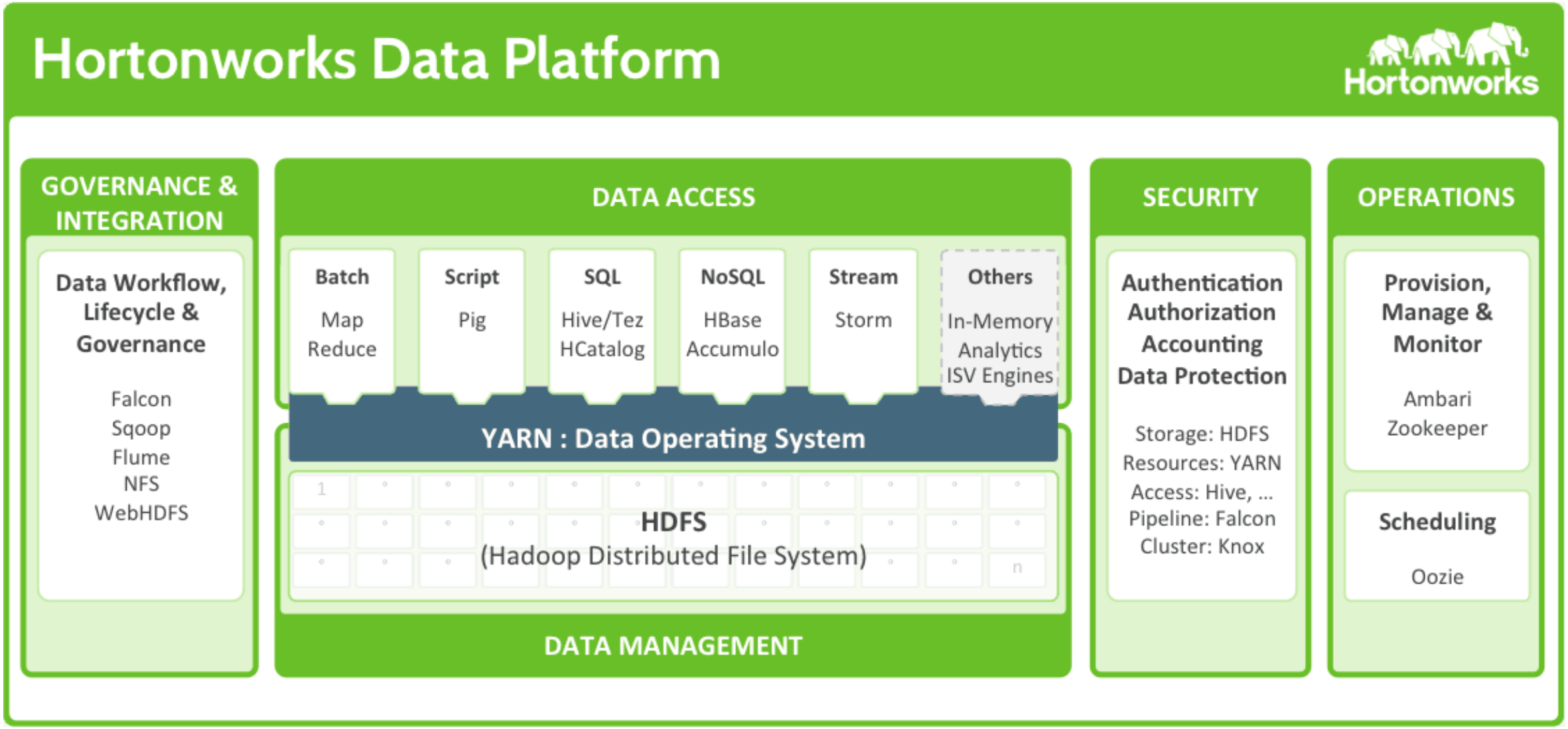
```
import org.apache.spark.rdd.RDD
import org.apache.spark.mllib.linalg.{Vector, Vectors}
import org.apache.spark.mllib.regression.LabeledPoint
import org.apache.spark.mllib.clustering.{KMeans, KMeansModel}
import lire.Base64ImageConverter
import lire.LireToolWrapperGF


val sqlContext = new org.apache.spark.sql.SQLContext(sc)
import sqlContext.implicits._


// FEATURE EXTRACTION USING OPEN SOURECE LIBRARIES ...


// THIS IS ONE SLICE OF THE "COMPOUND DATASET": Images can be vectorized in multiple ways, e.g., using color histograms
val dataPath = "/GITHUB/finance/data/out/image-analysis/A_2018-09-10-21-11-49/image_MD_ALL.parquet"

val df = sqlContext.read.parquet(dataPath)
val feature = df.select( "FCTH" )
val input = feature.rdd.map( x => x.getAs[org.apache.spark.mllib.linalg.Vector]("FCTH") );

val kmeans = new KMeans().setK(8).setSeed(1L)
val model = kmeans.run( input )

val WSSSE = model.computeCost(vectors)
println(s"Within Set Sum of Squared Errors = $WSSSE")

// Shows the result: TODO: WRAP THIS MODEL INTO A "TENZING LABELER WHICH ALSO PERSISTD THE RESULTS IN MD-Collection"
println("Cluster Centers: ")
model.clusterCenters.foreach(println)


val dataToCluster = ....
val labeledDdata = model.predict( dataToCluster )
```



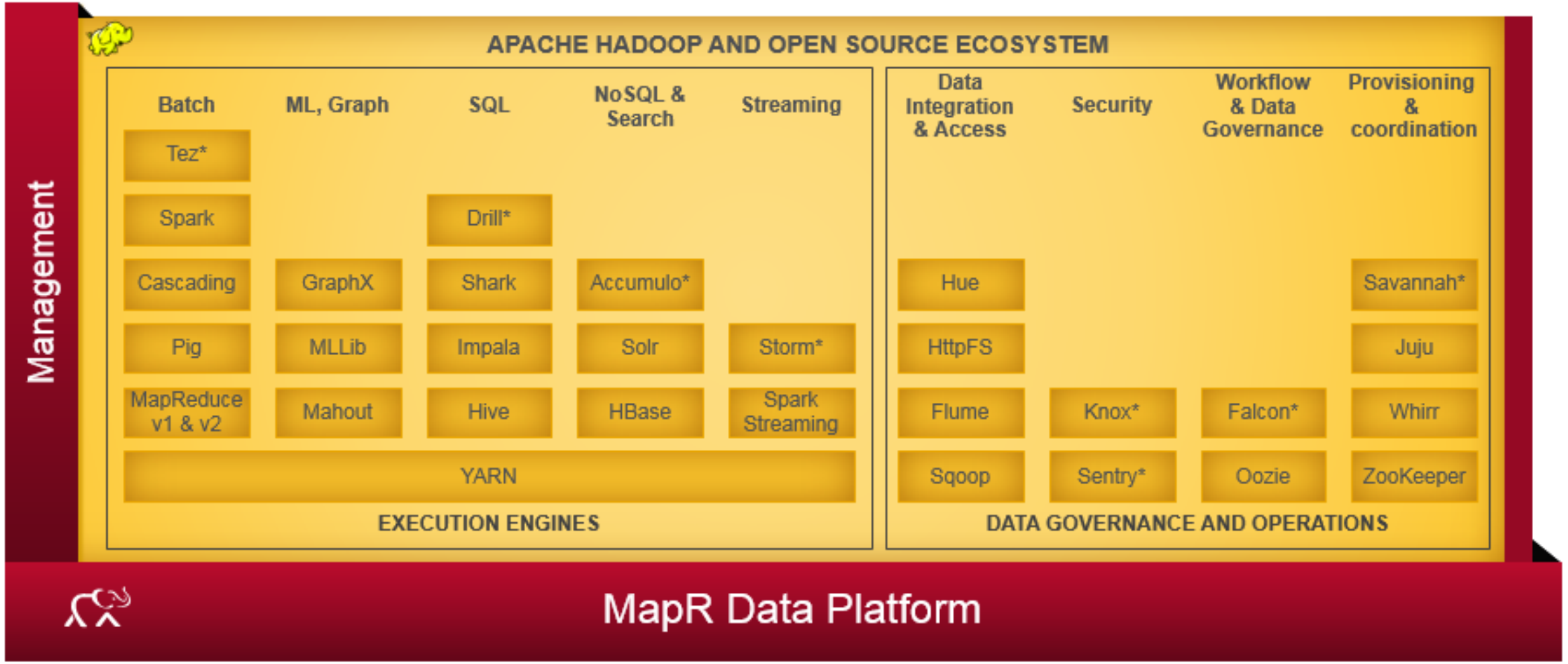
OUTLOOK: SPARK IMPLEMENTATION OF KMEANS CLUSTERING (SCALA)

```
def predict(pairs: Iterator[(String, (INDArray, Int, Int))]) = {
 val model = ClusteringModelSerializer.restoreKMeansModel(modelLoc)
 pairs.map{ case (name, imageFeatures) =>
        (name, run_kmeans_clustering_on_image_features(model, imageFeatures)
 }
}

def main(args: Array[String]) = {
 val sc = SparkContext(conf=conf)
 val hbase_io = common.HbaseIO(args)
 val out_format = common.KMeansOutputFormatter(args)

 val hbase_image_features = hbase_io.load_features_from_hbase(sc)
 val clustered_images = hbase_images.mapPartitions(predict) \
   .map(out_format.imagenet_format)
 val clustered_images.foreachPartition(hbase_io.put_to_hbase)
}
```

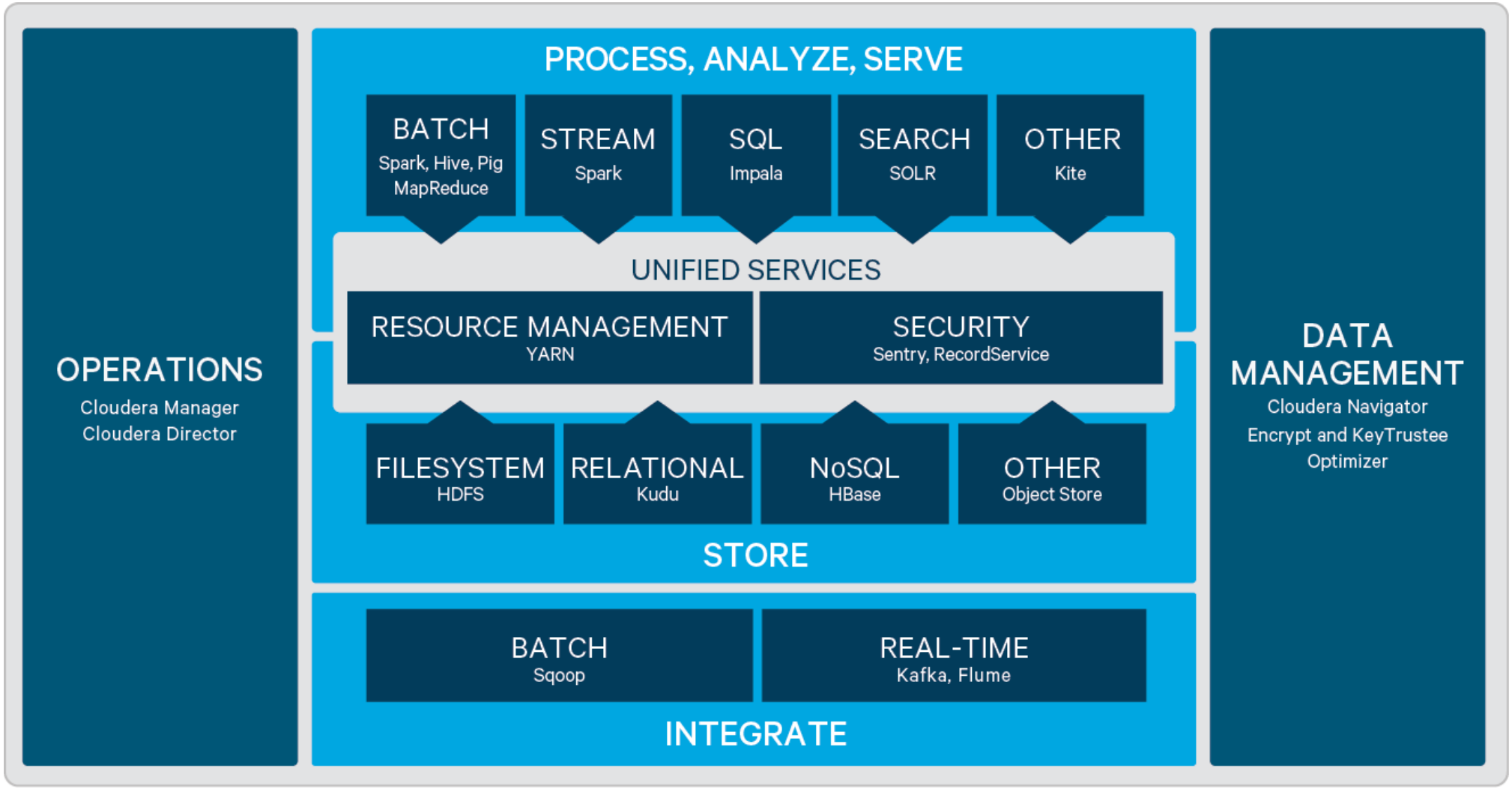
HortonWorks Architecture



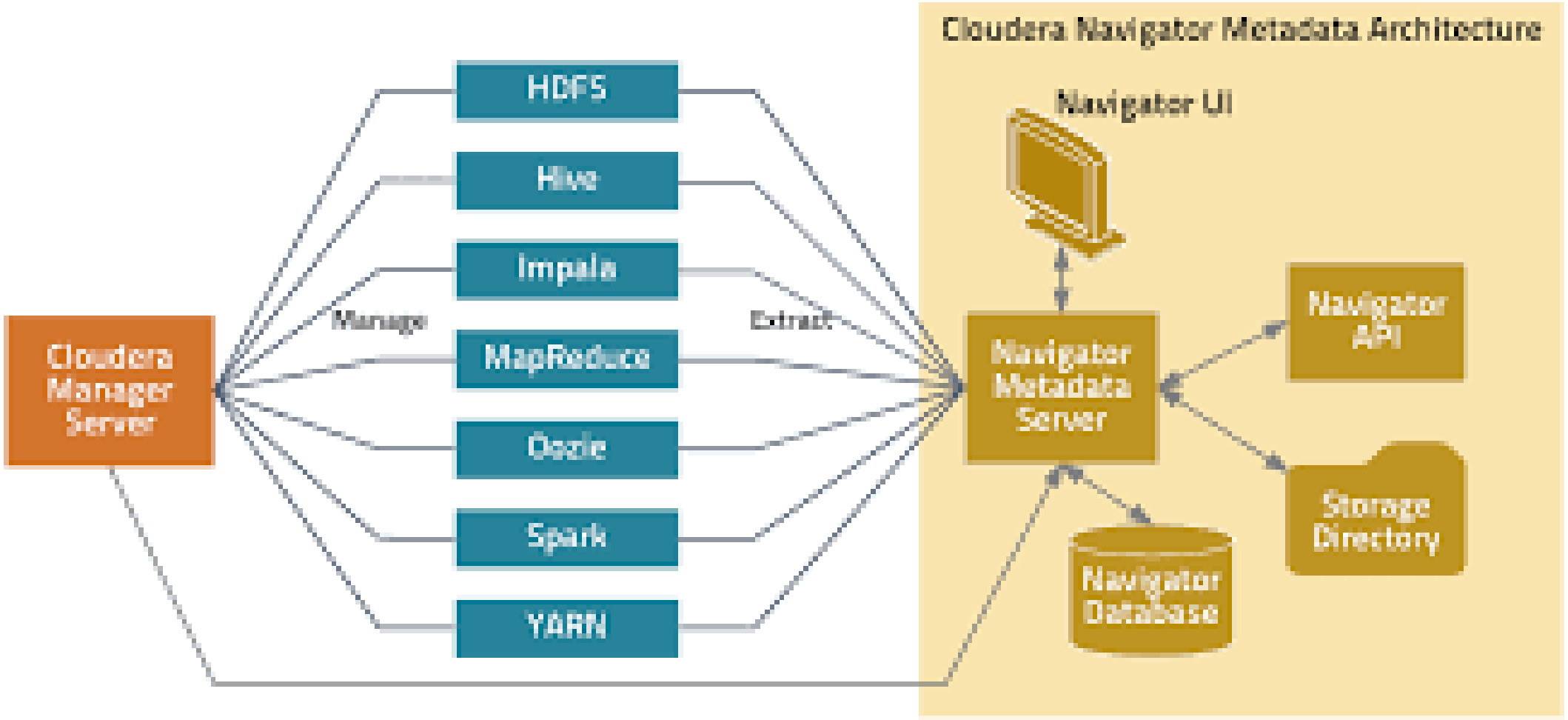
MapR Architecture



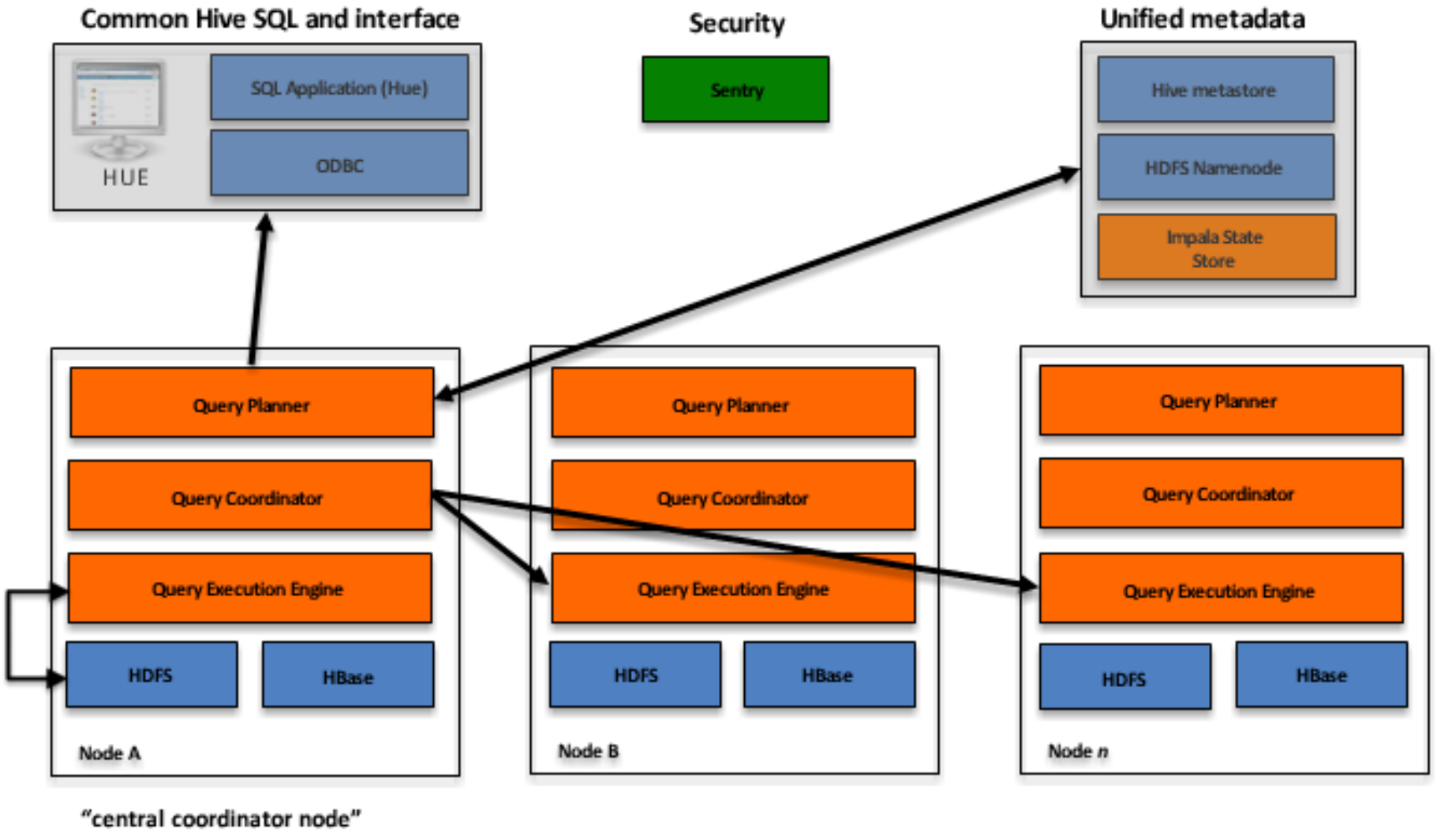
Cloudera CDH



Cloudera Architecture



Impala Architecture



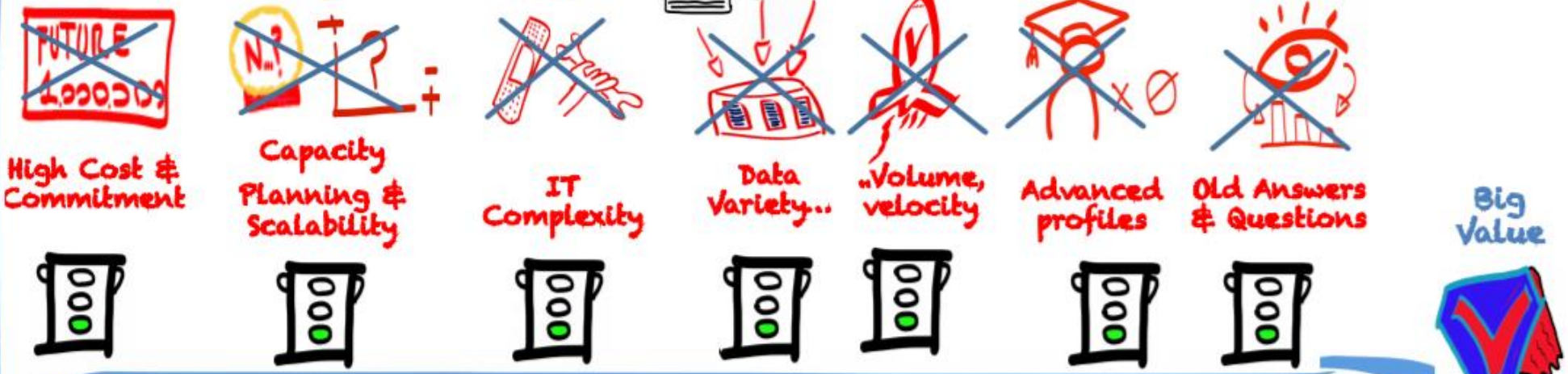
ie
SCHOOL OF
HUMAN SCIENCES
& TECHNOLOGY

BigData on AWS

https://aws.amazon.com/es/big-data/getting-started/

ie
SCHOOL OF HUMAN SCIENCES & TECHNOLOGY

Big Data Challenges:



BIG DATA Platform:

Lower Cost, OpEx

Experiment & learn more

Virtually unlimited & Elastic Resources

No heavy lifting & Reduced Time to Market, parallel processing on demand

Managed Services

Fully managed, secured & automated services that brings agility & focus

S3, EMR, Kinesis, Redshift, DynamoDB:

Collect all data, do Complex computations and processing it, both in Real-Time & Batch

Machine Learning

Easy deployment of ML powerful models without the need of ML Experts ready to be used

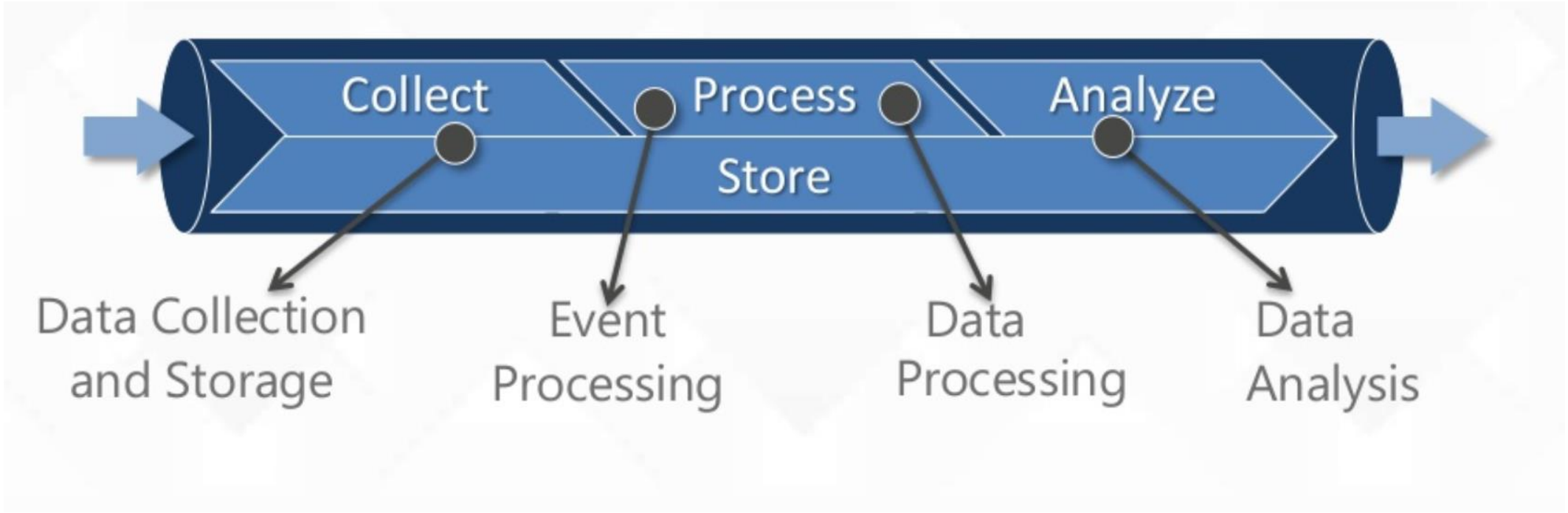
New Answers/ questions & Business Ideas

Extract the meaning from all your data & focus on new business Ideas, Models, etc..

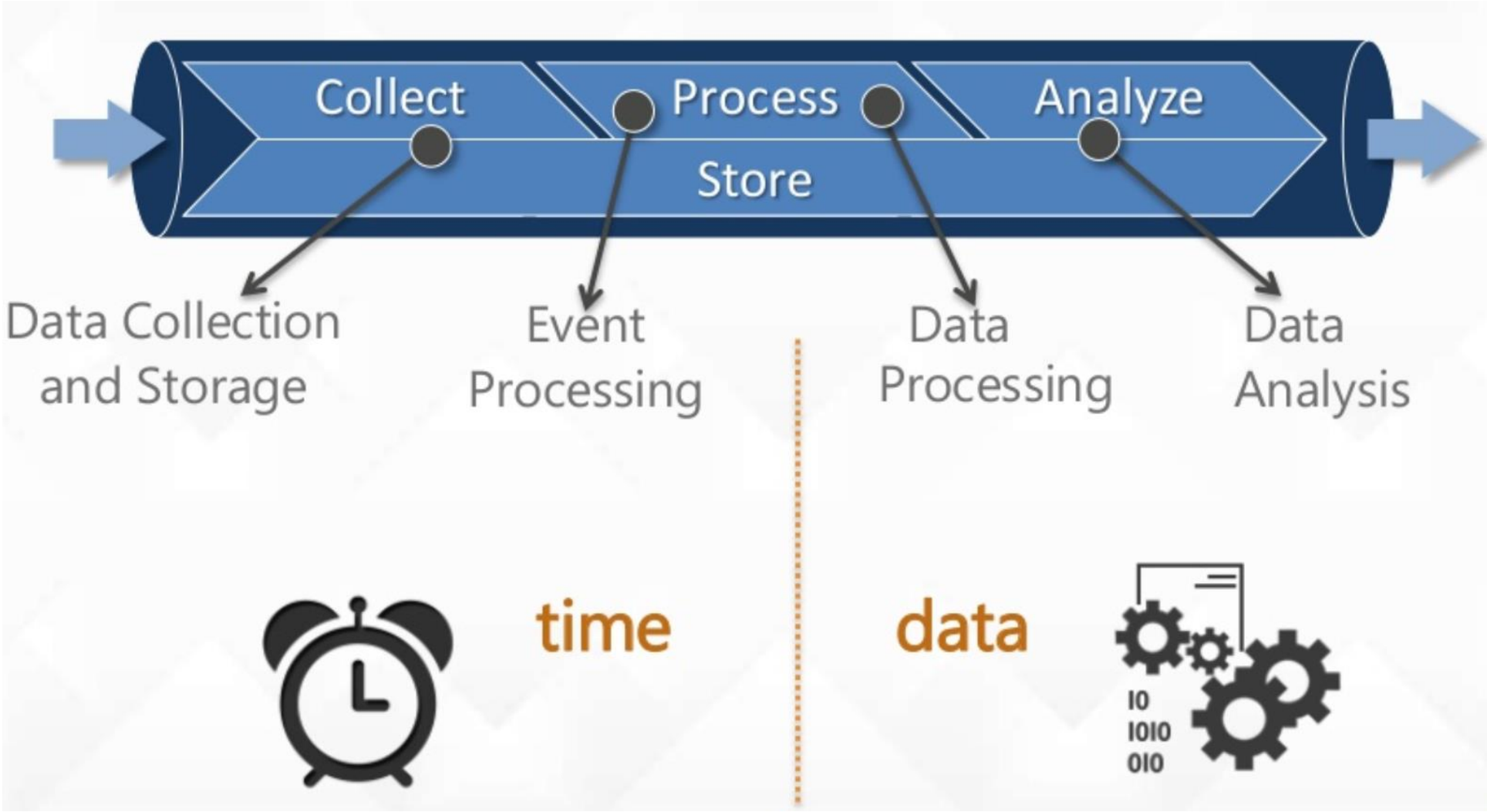
BigData Portfolio on AWS

Big Data Pipeline

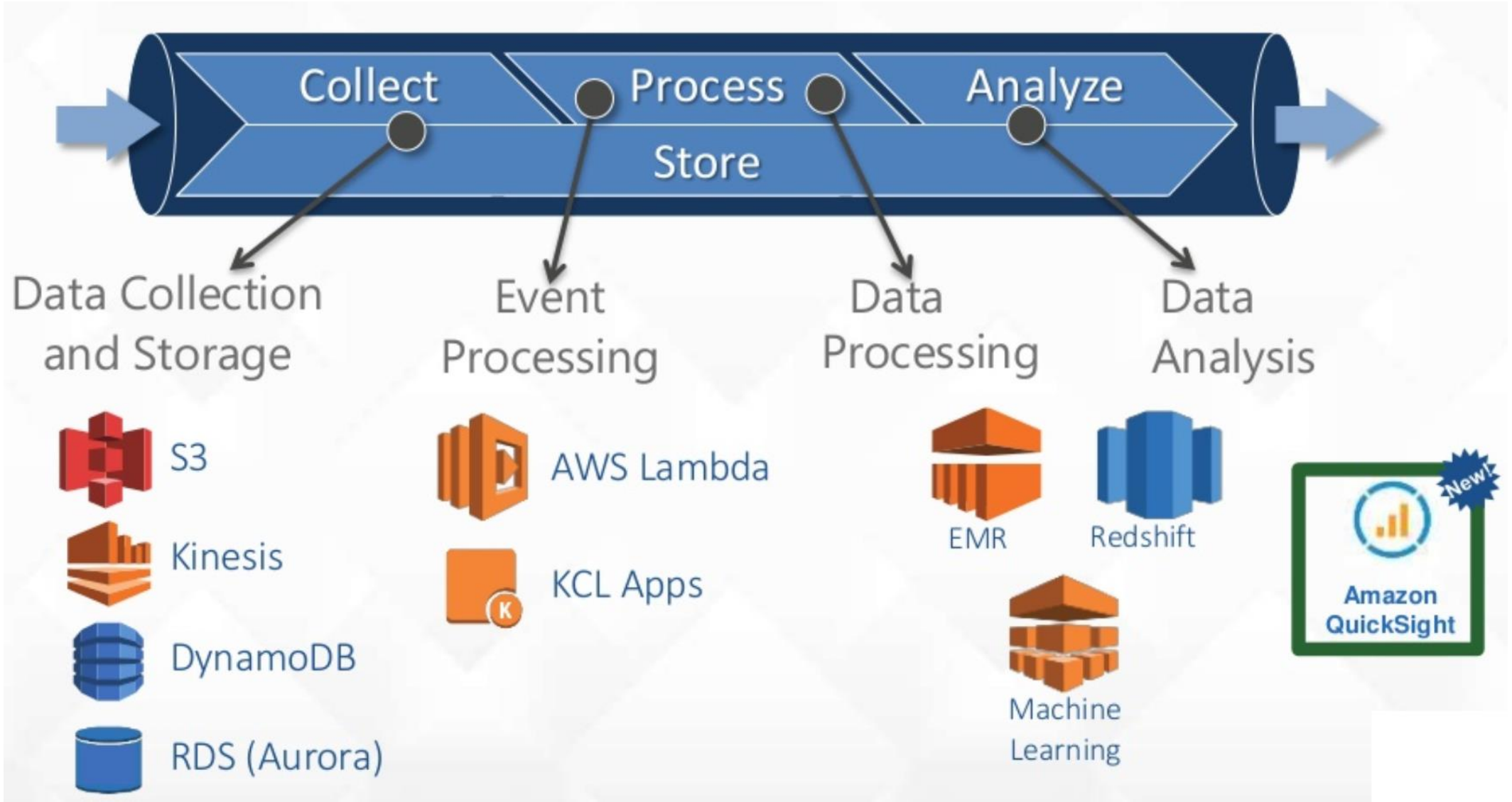
Primitive patterns



Primitive patterns

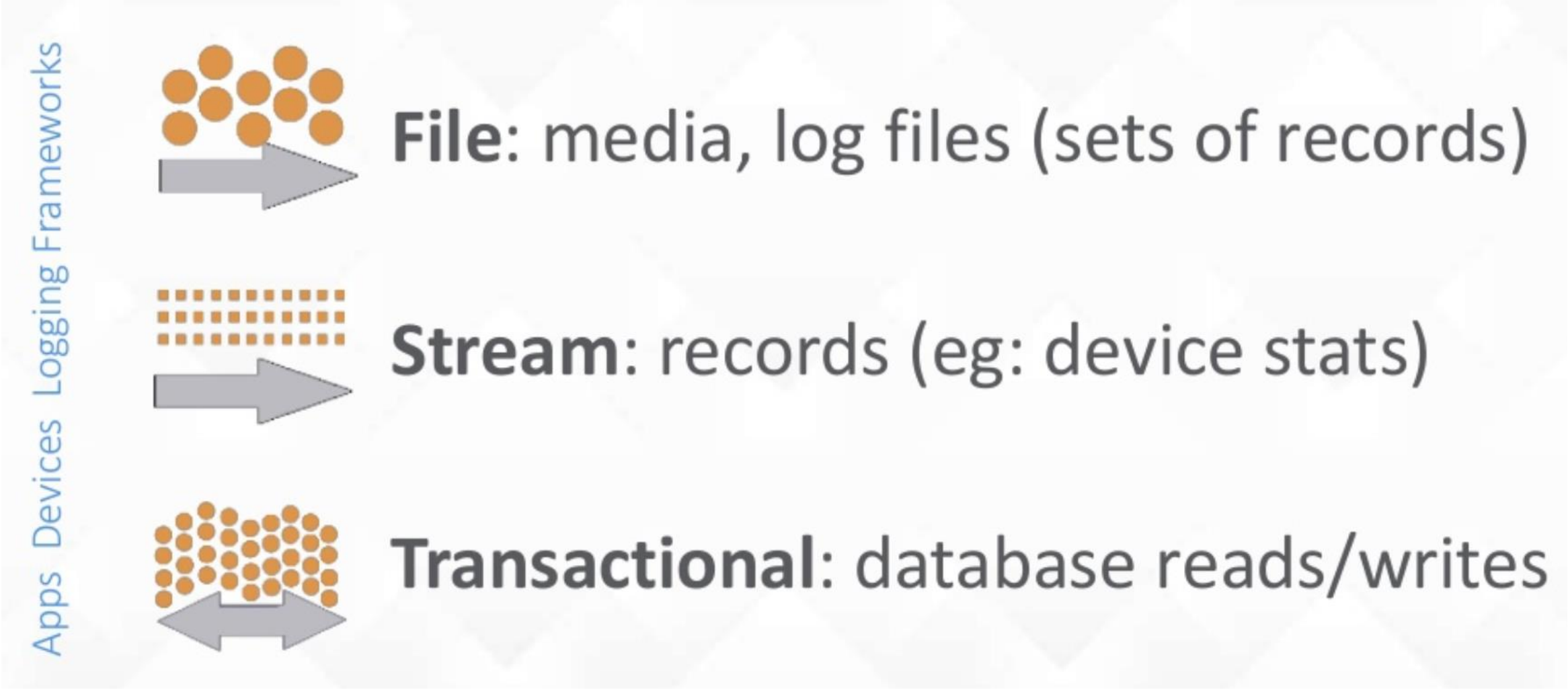


Primitive patterns



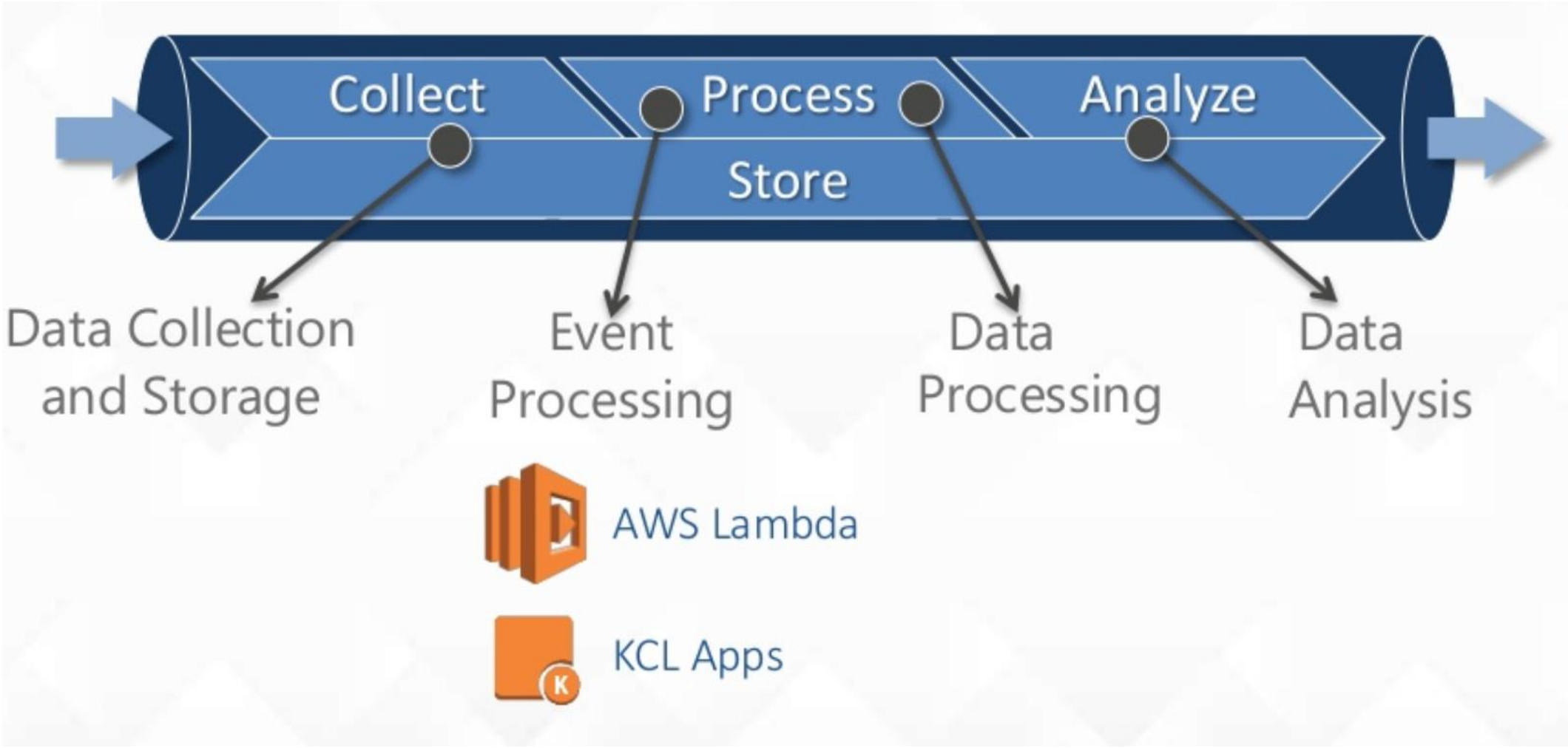
Primitive patterns

Data Collection and Storage

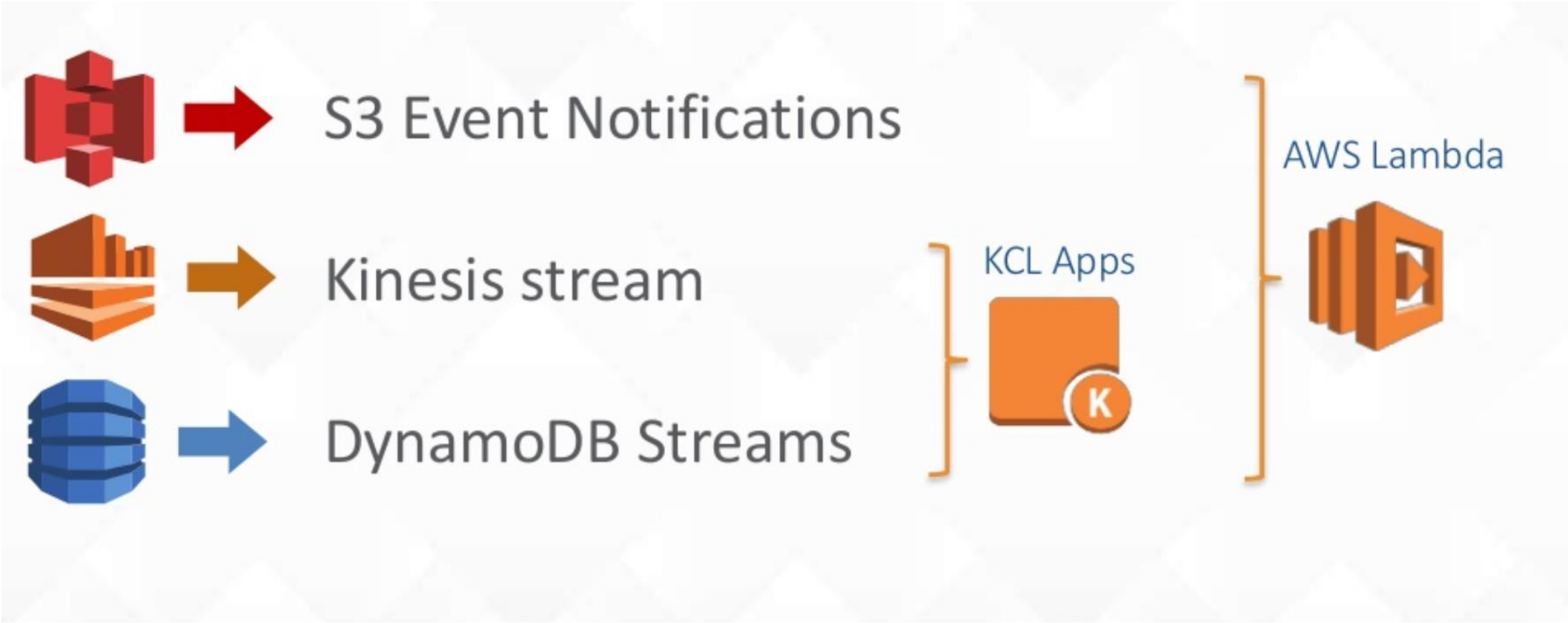


AWS Service: Data Collection and Storage

Primitive patterns

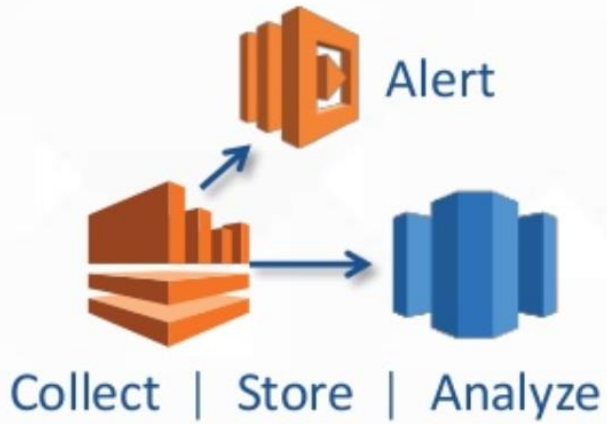


Event processing: enabling capabilities

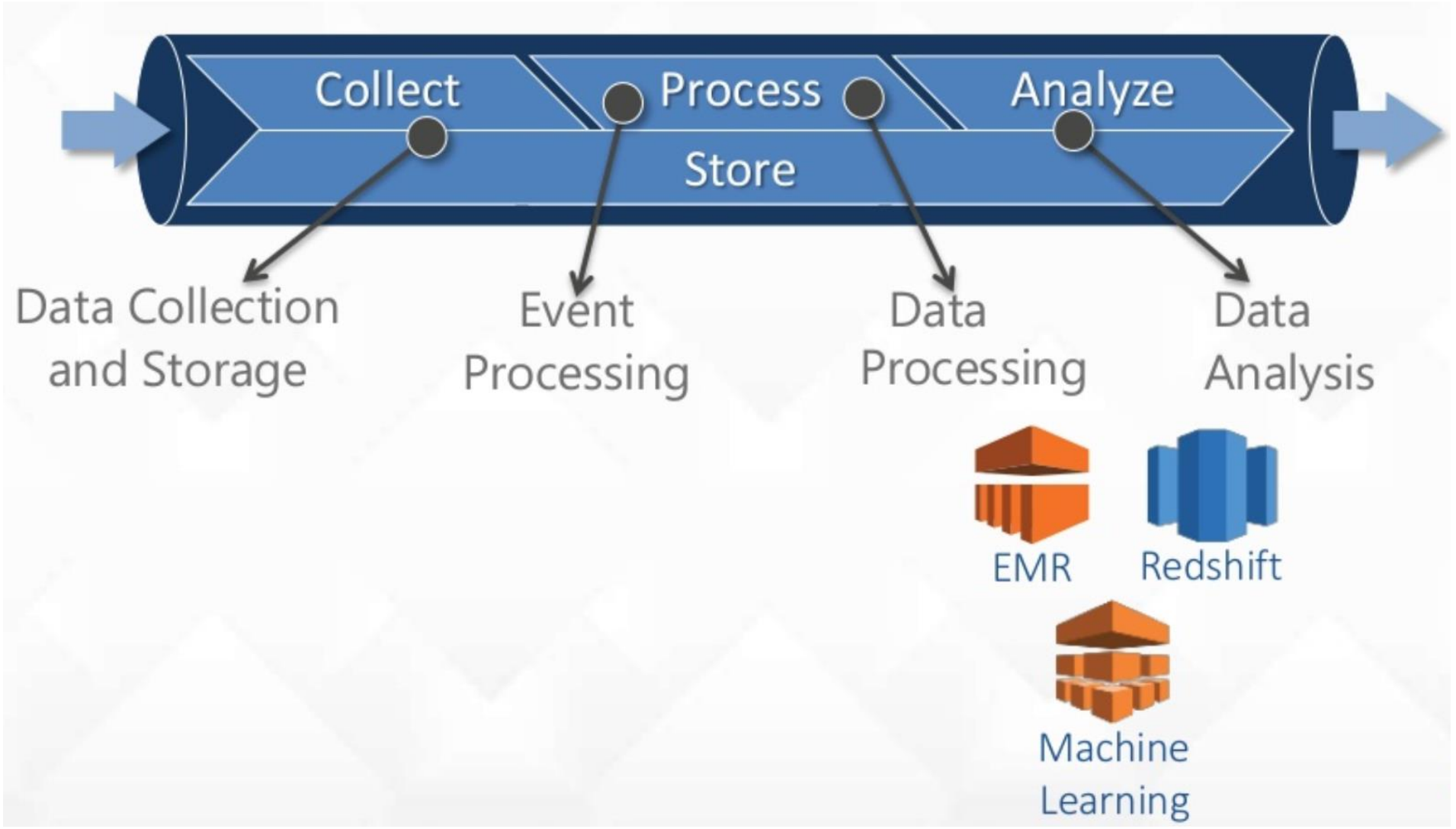


Benefits of Event Processing

- Build / add real-time events
  - Take action between data collection and analytics
    - Alerts and notifications, performance and security
    - Automated data enrichment (eg: aggregations)
- De-couple application modules
  - Streamline development and maintenance
  - Increase agility
    - MVP + iterate on discrete components



Primitive patterns



ie
SCHOOL OF
HUMAN SCIENCES
& TECHNOLOGY

BigData on Azure

https://azure.microsoft.com/en-us/solutions/big-data/

Hadoop Ecosystem

Ambari / Cloudera Manager / Cloudera Director
Cluster Management and Monitoring UI

Sqoop
RDBMS Exchange

Flume
Log Collector

Zookeeper
HA / Distributed Coordination of Resources

HIVE
SQL Language

Pig
Scripting / ETL

Spark
SQL / Streaming

Storm
Streaming

Impala
Fast SQL Querying

Oozie
Workflow / Automation

Mahout
Machine Learning

Solr
Search

HBASE
Columnar Store

YARN (MapReduce V2)
Yet Another Resource Negotiator / Manager

HDFS
Hadoop Distributed File System

Hadoop On Your Terms

[100% Apache Hadoop-based Service in the Cloud]

[Hortonworks Data Platform is now Microsoft Azure Certified]

[Cloudera Selects Microsoft Azure as a Preferred Cloud Platform]

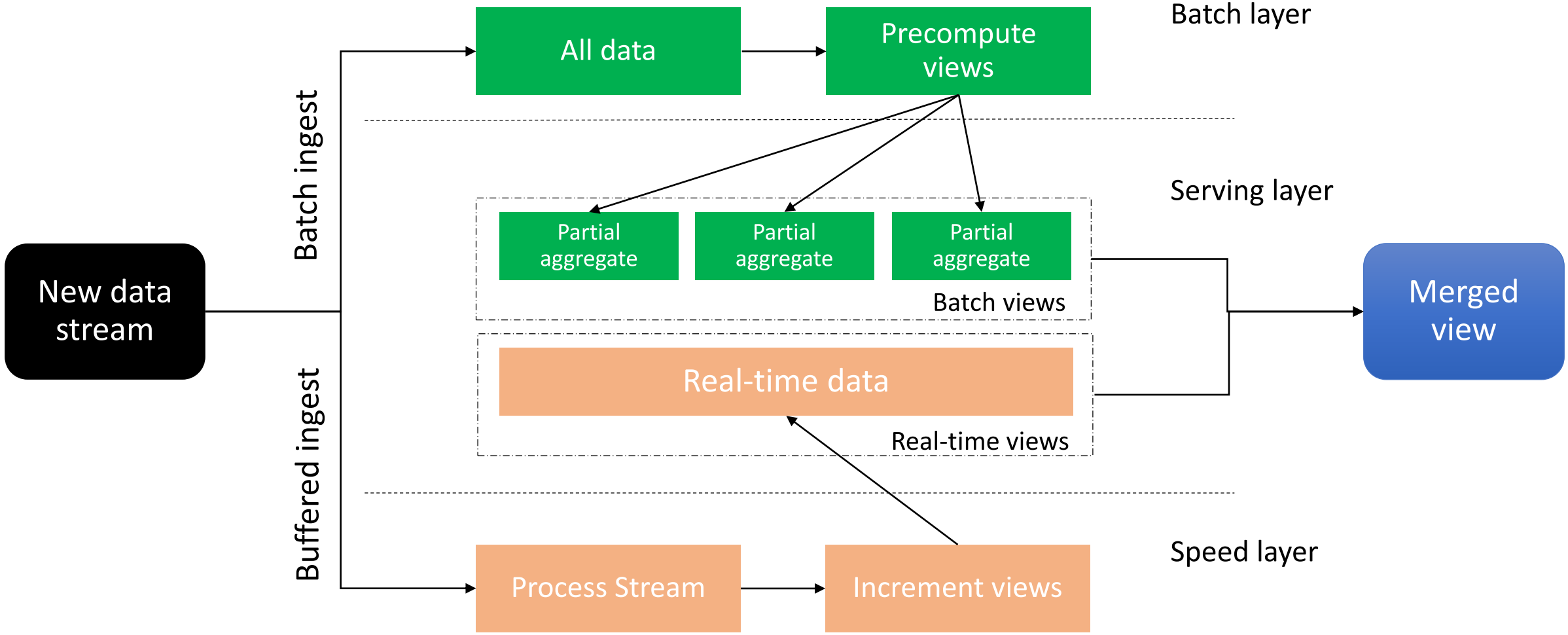
Allied Market Research estimates the global Hadoop market to grow from $2 billion in 2013 to $50.2 billion by the year 2020.

Azure HDInsight

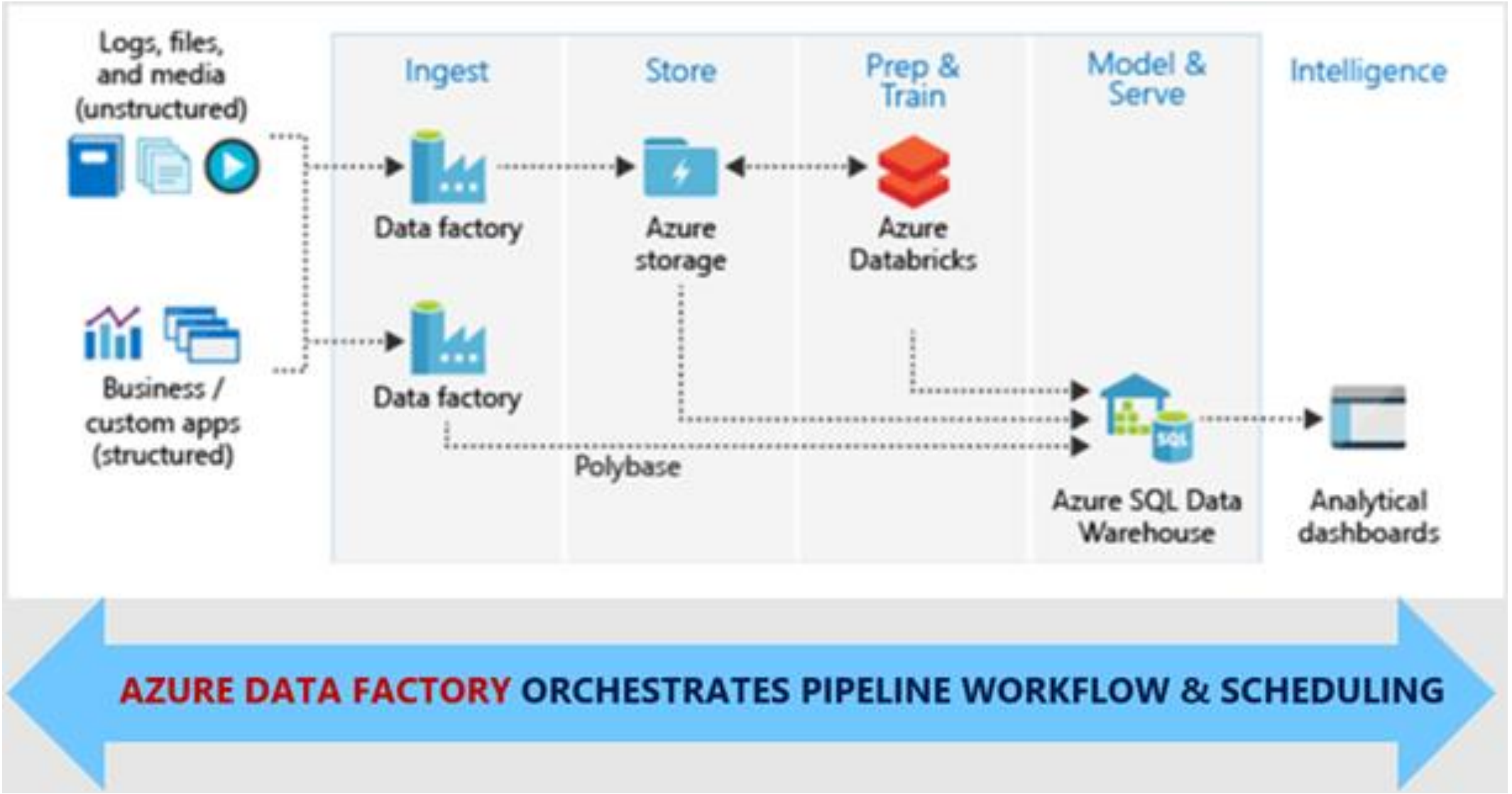
Hadoop in Azure

- Massively Parallel Processing (MPP) managed Hadoop cluster
- Spark SQL for real-time queries from in-memory caching
- Storm and Spark Streaming for near real-time processing of IoT
- HBase as a columnar NoSQL transactional database
- Hive SQL queries for batch processing

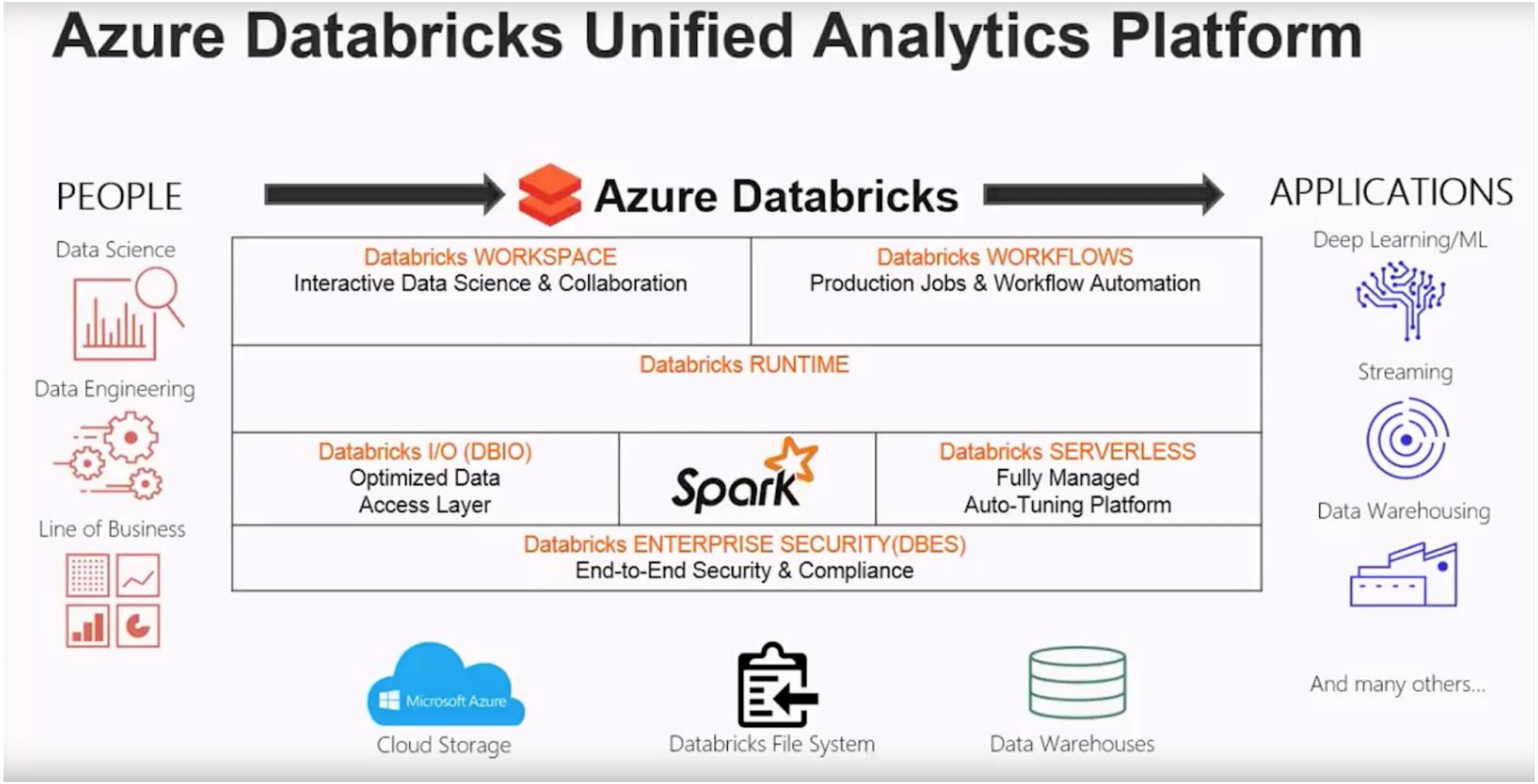
Lambda Architecture – High Level View



Azure Databricks



Azure Databricks

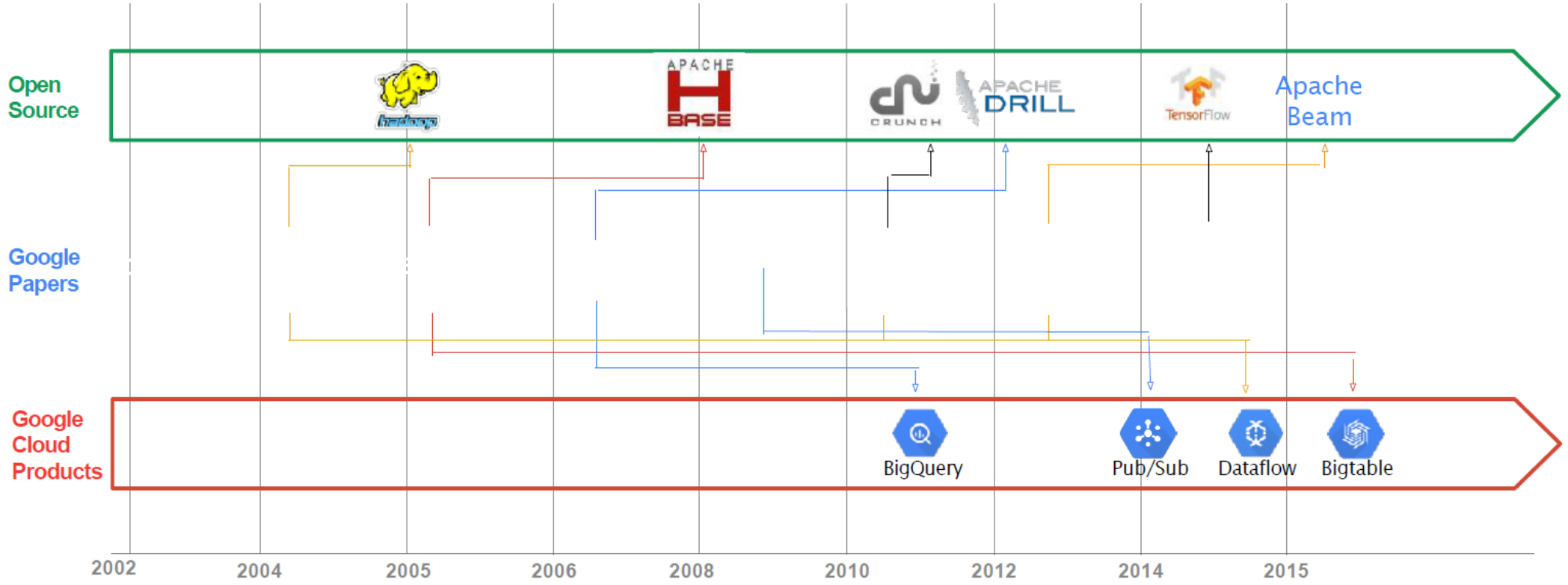


ie
SCHOOL OF
HUMAN SCIENCES
& TECHNOLOGY

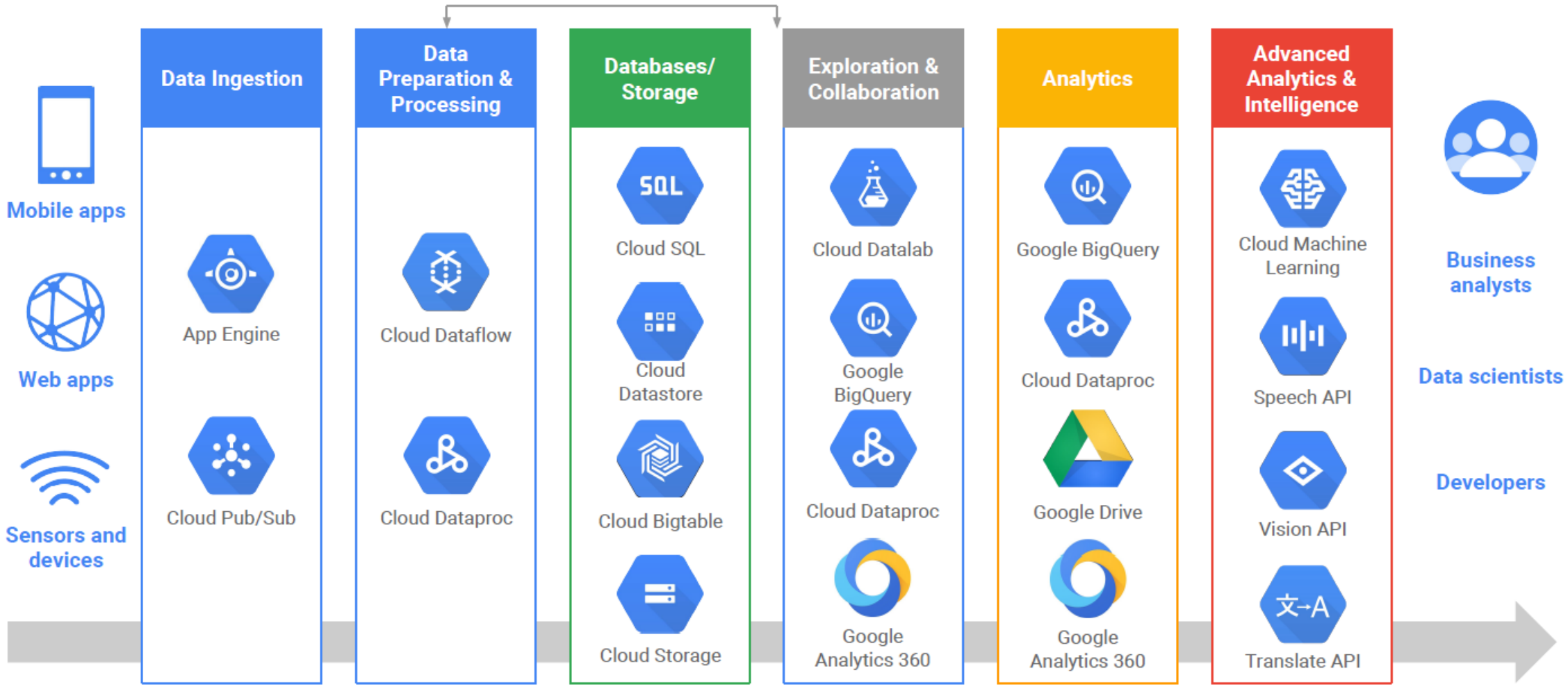
BigData on Google Cloud

https://cloud.google.com/solutions/big-data

10 years of Tackling Big Data Problems

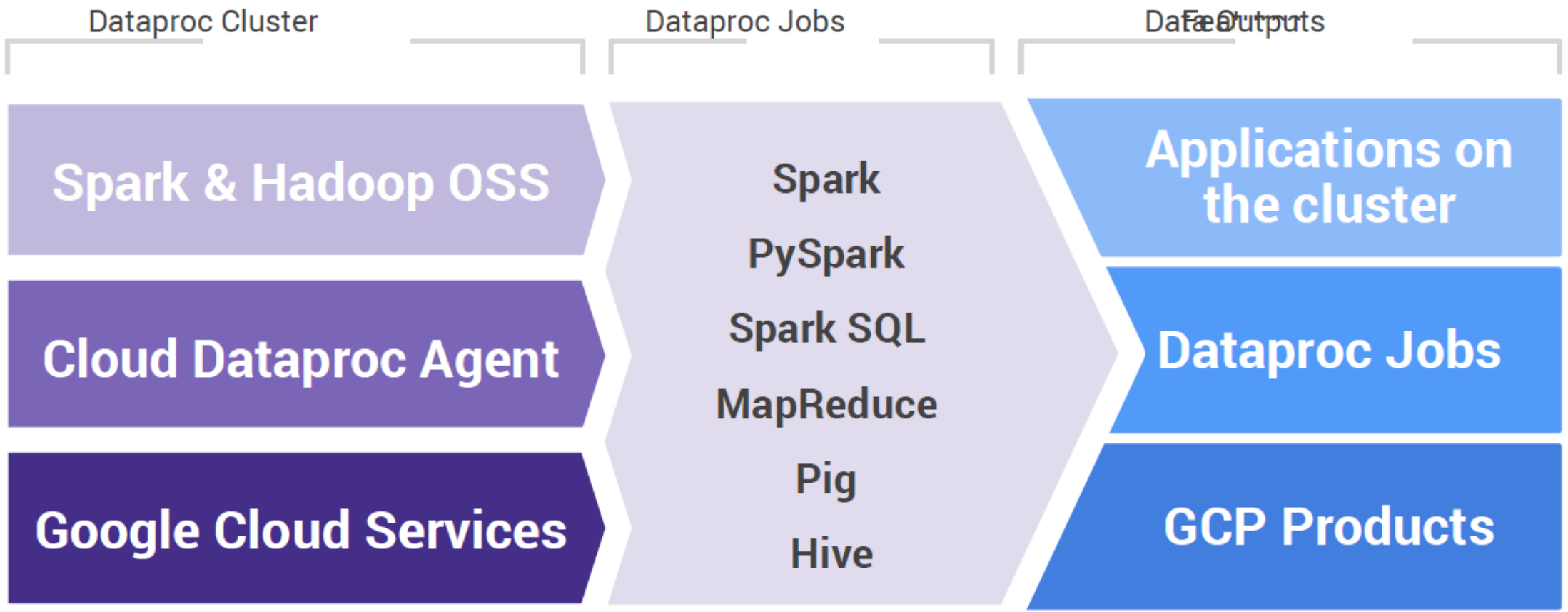


Transform Data into Actions



Google Cloud DataProc

- Apache Spark and Apache Hadoop should be **fast**, **easy**, and **cost-effective**

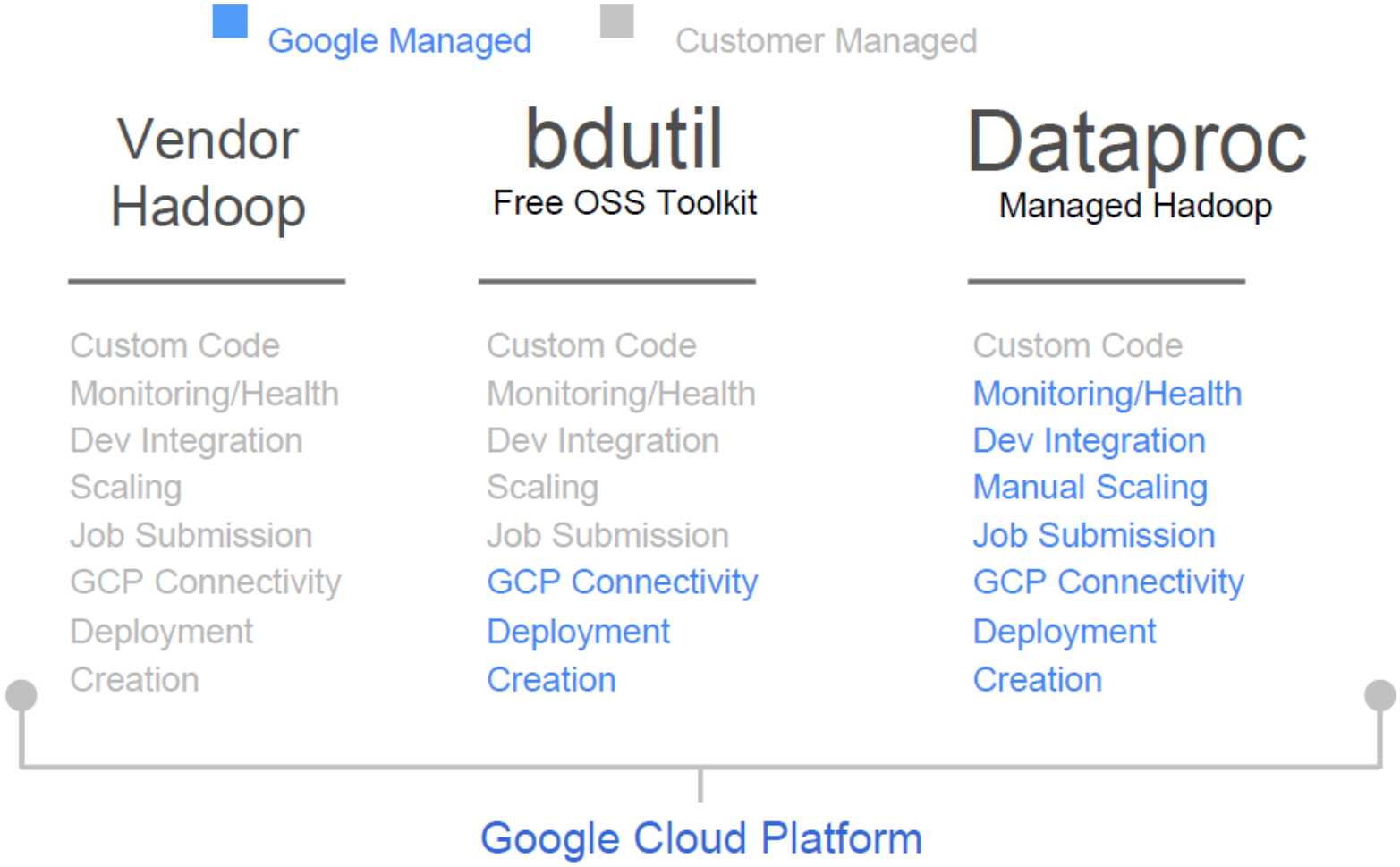


ie
SCHOOL OF HUMAN SCIENCES & TECHNOLOGY

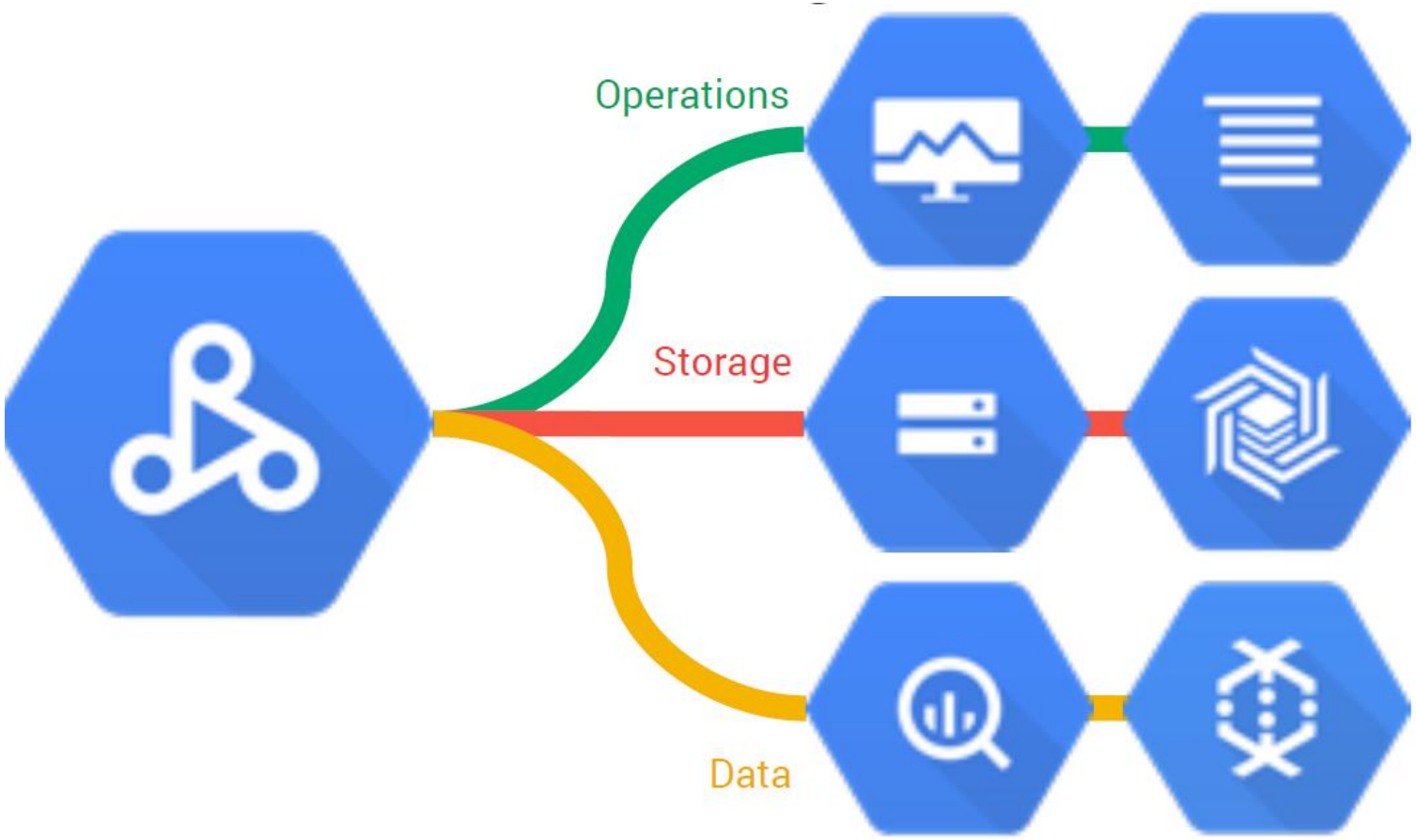
Easy, Fast. Cost-effective

- **Fast**
  - Things take seconds to minutes, not hours or weeks
- **Easy**
  - Be an expert with your data, not your data infrastructure
- **Cost-effective**
  - Pay for exactly what you use

Running Hadoop on Google Cloud

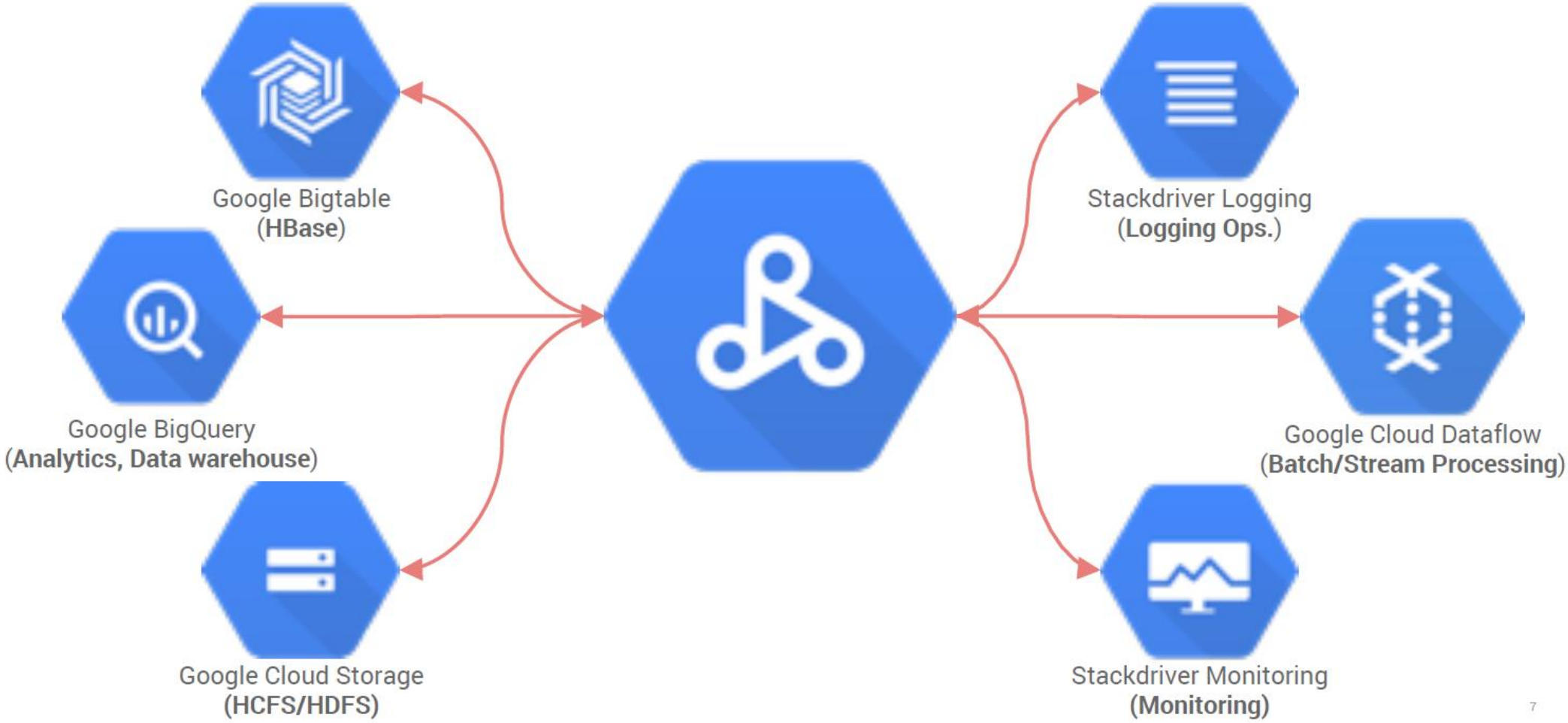


Cloud DataProc - **integrated**



Cloud Dataproc is natively integrated with several Google Cloud Platform products as part of an **integrated data platform**.

Where Cloud DataProc fits into GCP



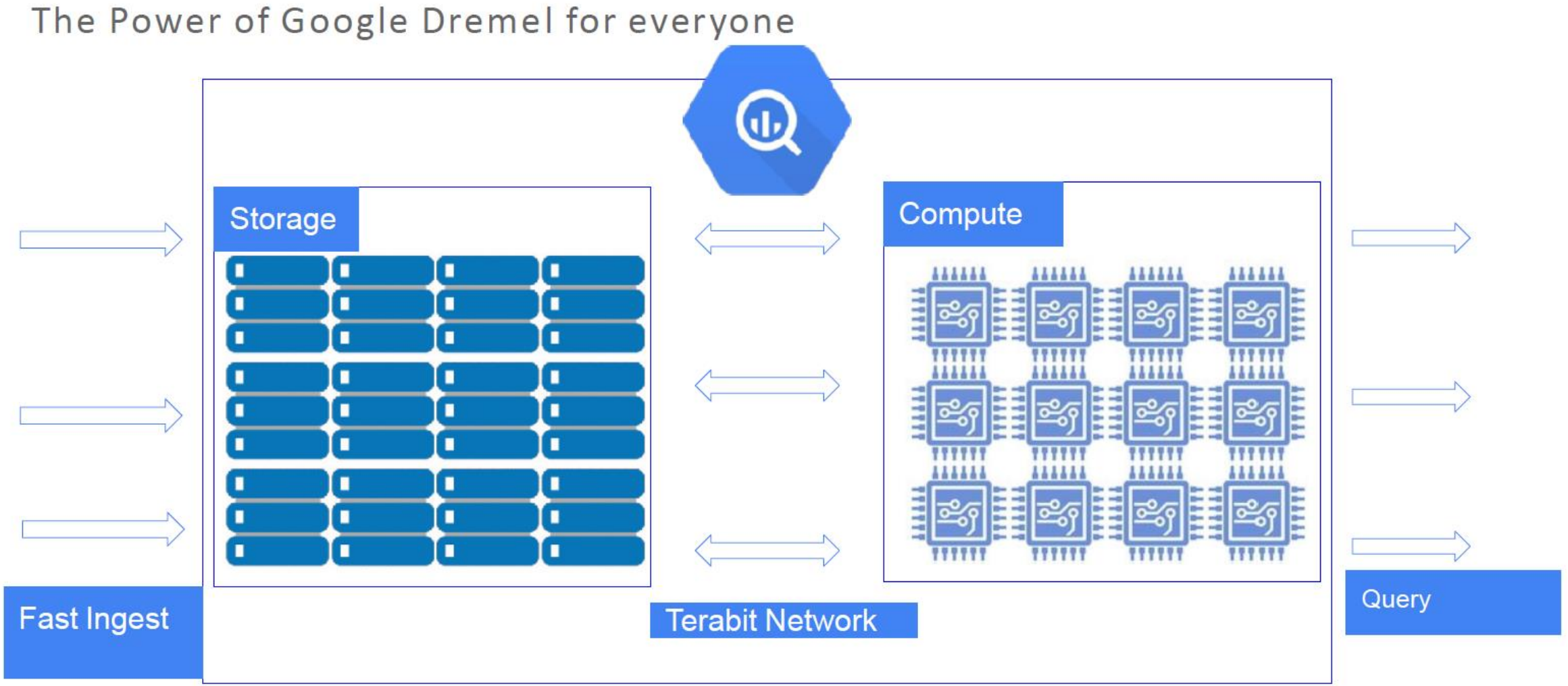
Google BigQuery

- makes complex data analysis simple
  - Scales automatically
  - No setup or administration
  - Stream up to 100,000 rows p/sec
  - Easily integrates with third-party software

Running an *inefficient* regular expression over 100 billion rows in

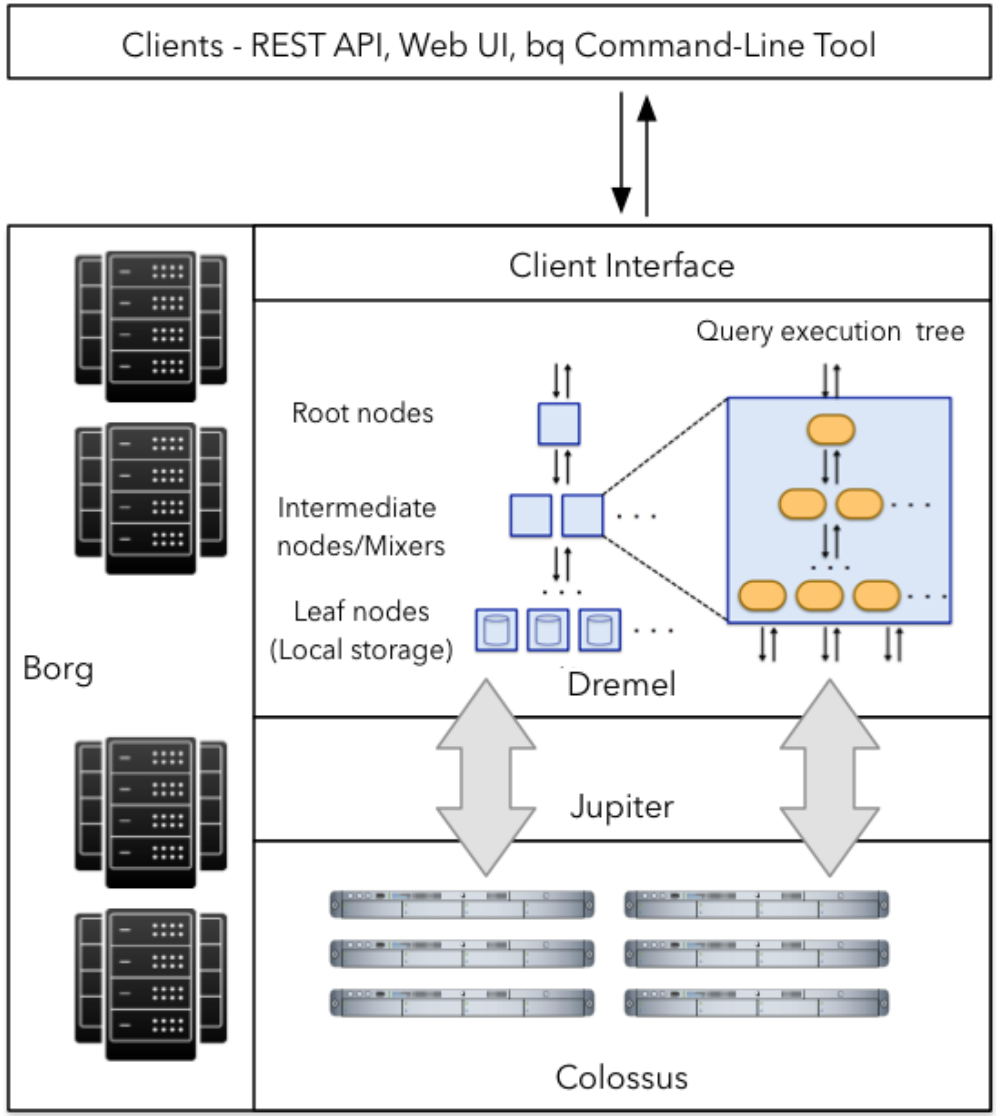
less than 60 seconds

Google BigQuery Architecture

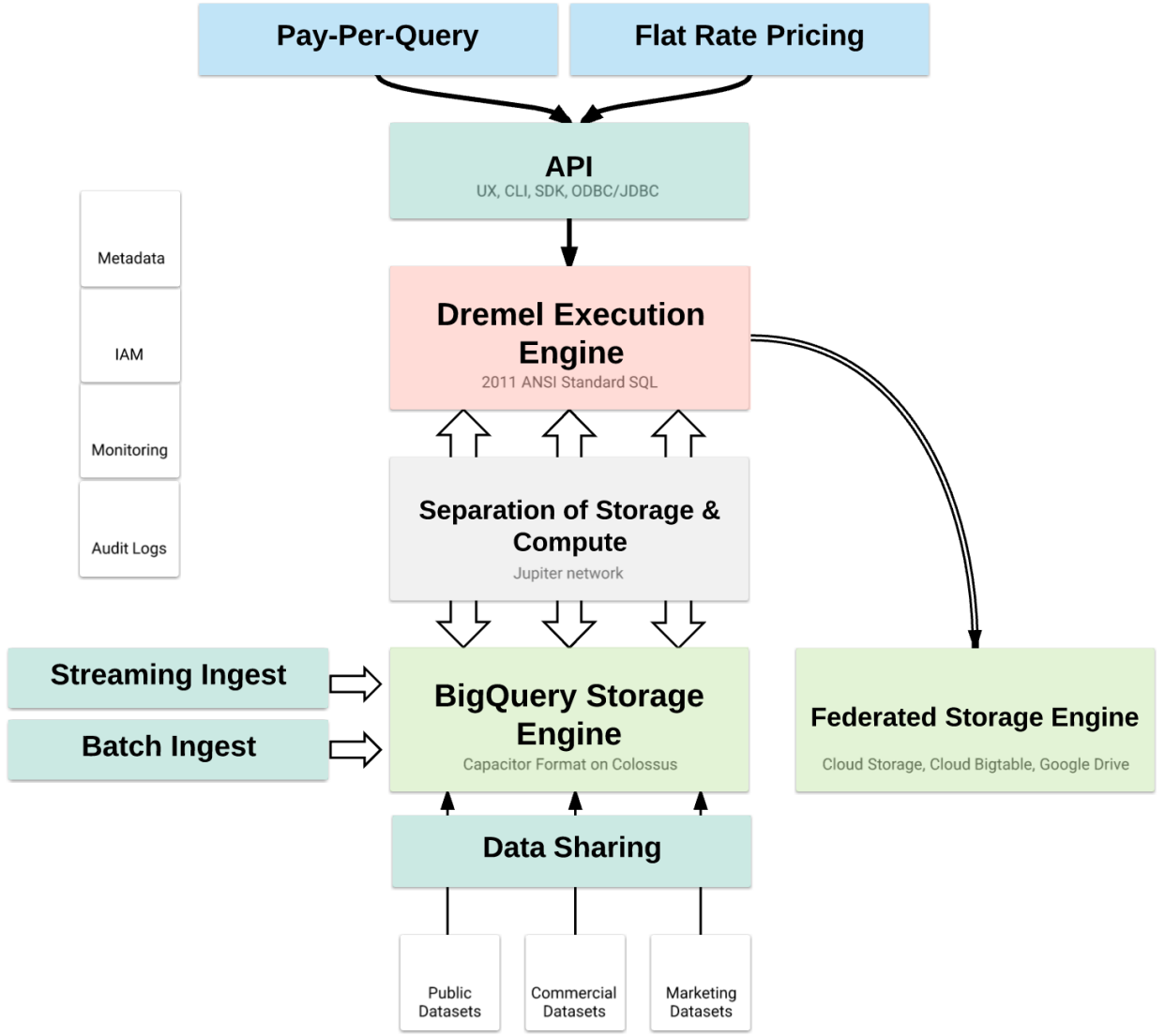


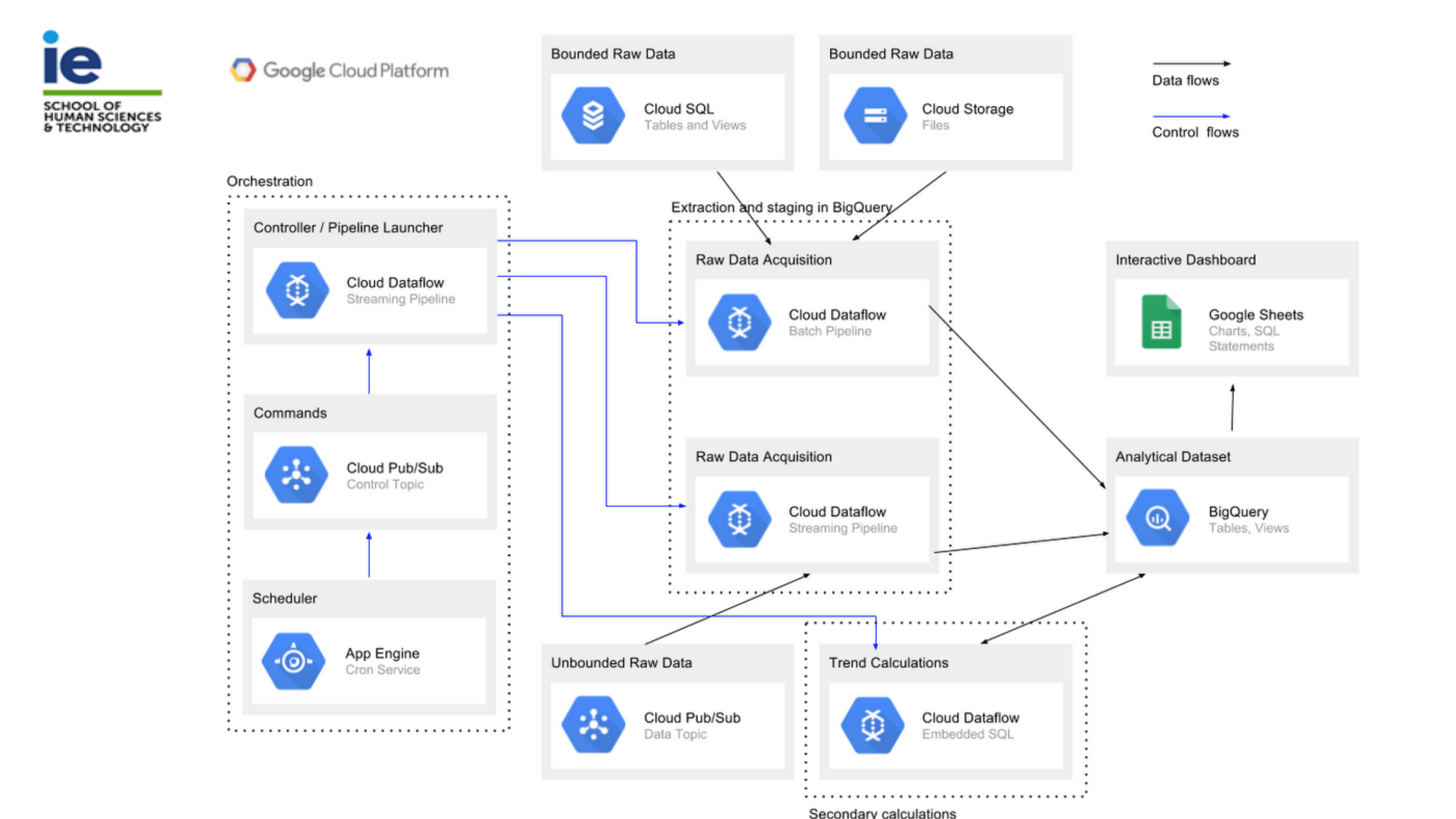
Google BigQuery Architecture

- Google Dremel
- Google Jupiter
- Google Colossus



Google BigQuery Architecture




SCHOOL OF HUMAN SCIENCES & TECHNOLOGY
Google Cloud Platform
Bounded Raw Data
Cloud SQL
Tables and Views
Bounded Raw Data
Cloud Storage
Files
Data flows
Control flows
Orchestration
Controller / Pipeline Launcher
Cloud Dataflow
Streaming Pipeline
Commands
Cloud Pub/Sub
Control Topic
Scheduler
App Engine
Cron Service
Extraction and staging in BigQuery
Raw Data Acquisition
Cloud Dataflow
Batch Pipeline
Raw Data Acquisition
Cloud Dataflow
Streaming Pipeline
Interactive Dashboard
Google Sheets
Charts, SQL Statements
Analytical Dataset
BigQuery
Tables, Views
Unbounded Raw Data
Cloud Pub/Sub
Data Topic
Trend Calculations
Cloud Dataflow
Embedded SQL
Secondary calculations

ie
SCHOOL OF
HUMAN SCIENCES
& TECHNOLOGY

BigData on Oracle Cloud

https://www.oracle.com/big-data/big-data-service/

OCI Hardware

Bare Metal Standard
52 Cores, 768 GB RAM,
up to 1 PB Block Storage

Bare Metal DenseIO
51.2 TB of local NVMe SSD
2x 25Gbe Network Interfaces

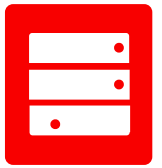
Bare Metal GPU
28 Cores, 192 GB RAM,
2x Tesla P100 GPUs
Pre-Configured Images

Bare Metal GPU V2
52 Cores, 768 GB RAM,
8x Tesla V100 GPUs
NVLINK Interconnect

File Storage Service
Managed distributed file service
POSIX, NFSv3 mount point

Block Storage
50 GB-2 TB volumes
Up to 400K IOPS per host

Bare-Metal HPC
36 cores, 3.7 GHz
384GB RAM
6.7 TB NVME, 1PB block
RDMA



GPU Visualization
P100 GPU
NVIDIA Quadro Enabled
Teradici & Citrix Support

The Best of Hadoop Ecosystem

There's no way around it. Hadoop is complicated. But Cloudera Enterprise Data Hub makes it easier to take advantage of Hadoop. With Oracle Big Data Service and Cloudera, you can:

Use all of Cloudera Enterprise Data Hub

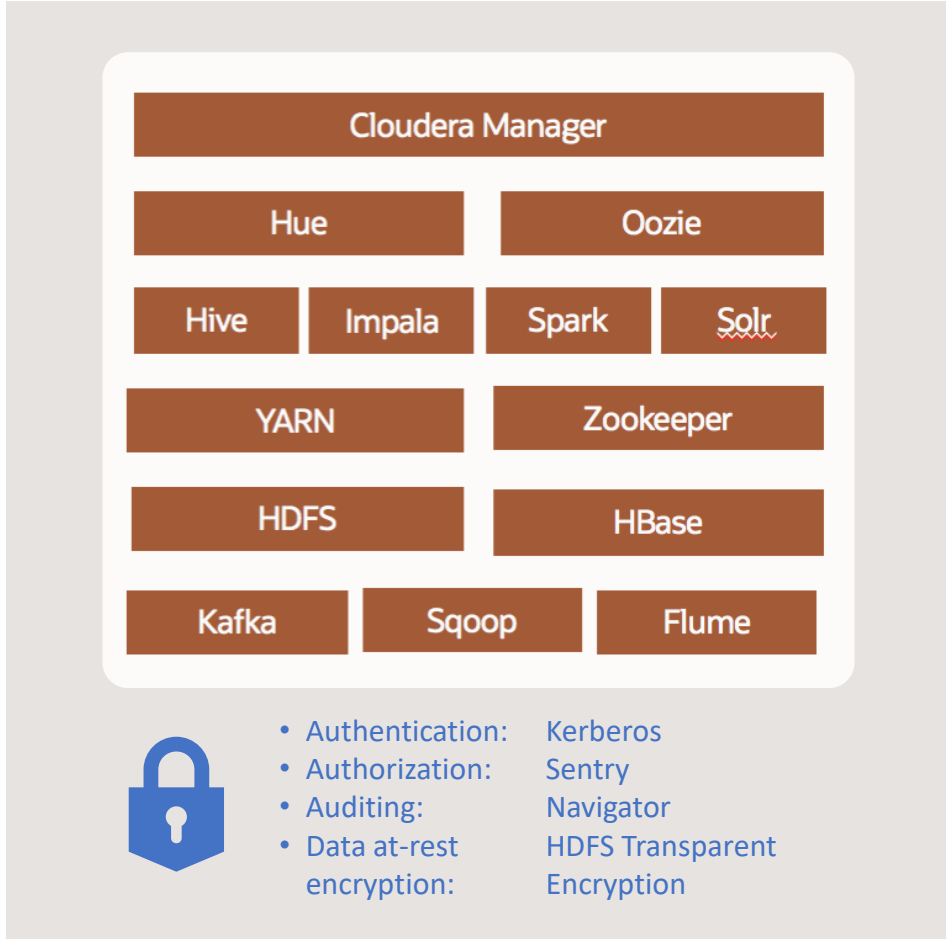
Have vertically integrated Spark, Kafka, Impala, Hive, Solr, and more

Have automatically deployed security and management features

You also can choose your Cloudera version, giving you the ability to:

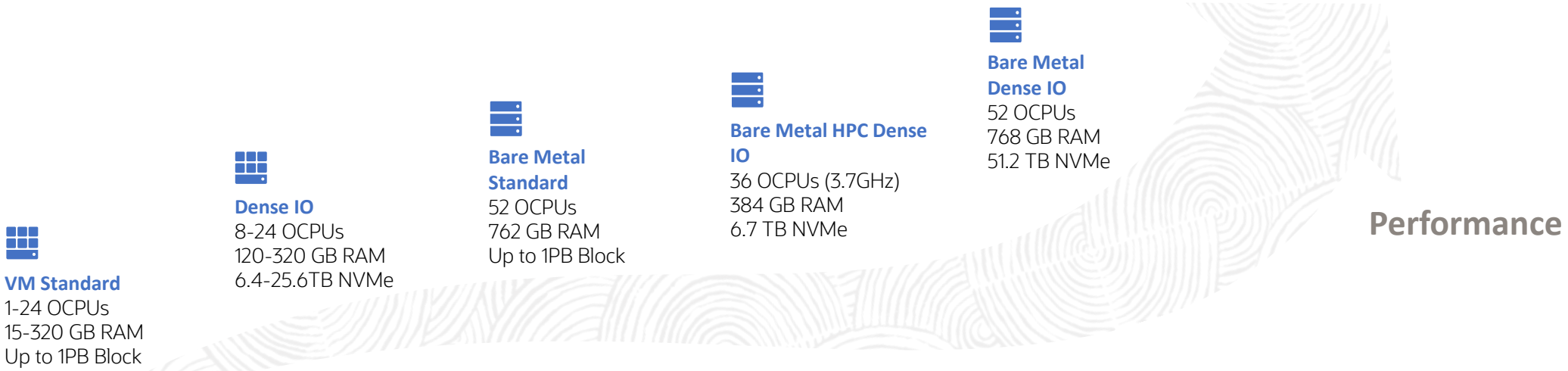
Match your current deployment
which is important for test and dev environments

Deploy new versions
allowing you to take advantage of the distribution's latest features



Oracle Big Data Service offers the compute and storage you want

Development → Test → Data Science → Data Lakes



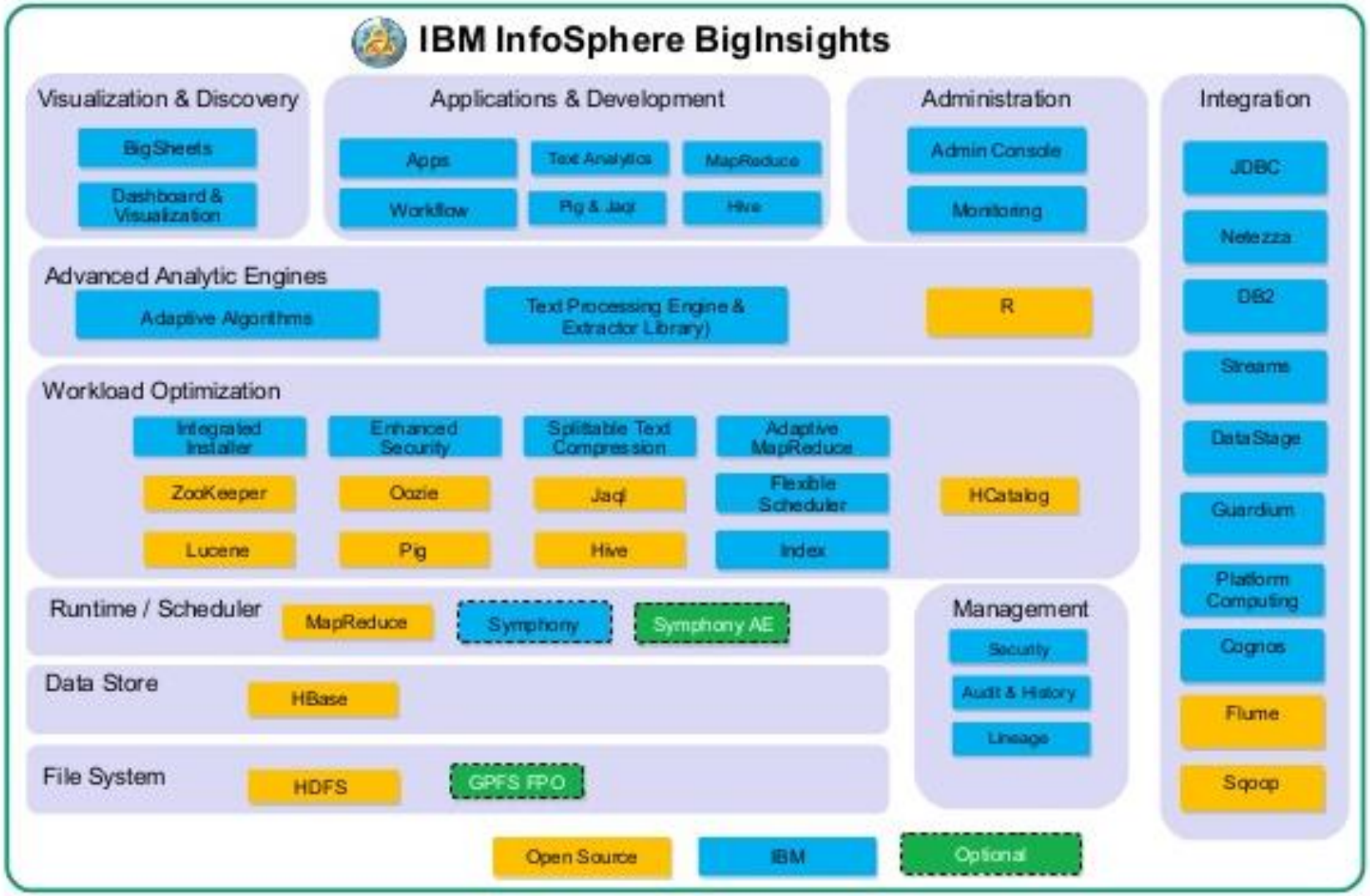
You can use Oracle Big Data Cloud for development, test, data science, or data lakes. In all of these cases, we provide the compute offerings you need for your use case. You can achieve ultimate flexibility by combining bare metal and virtual machines with different storage options, which optimizes your deployment based on your resource needs.

ie
SCHOOL OF
HUMAN SCIENCES
& TECHNOLOGY

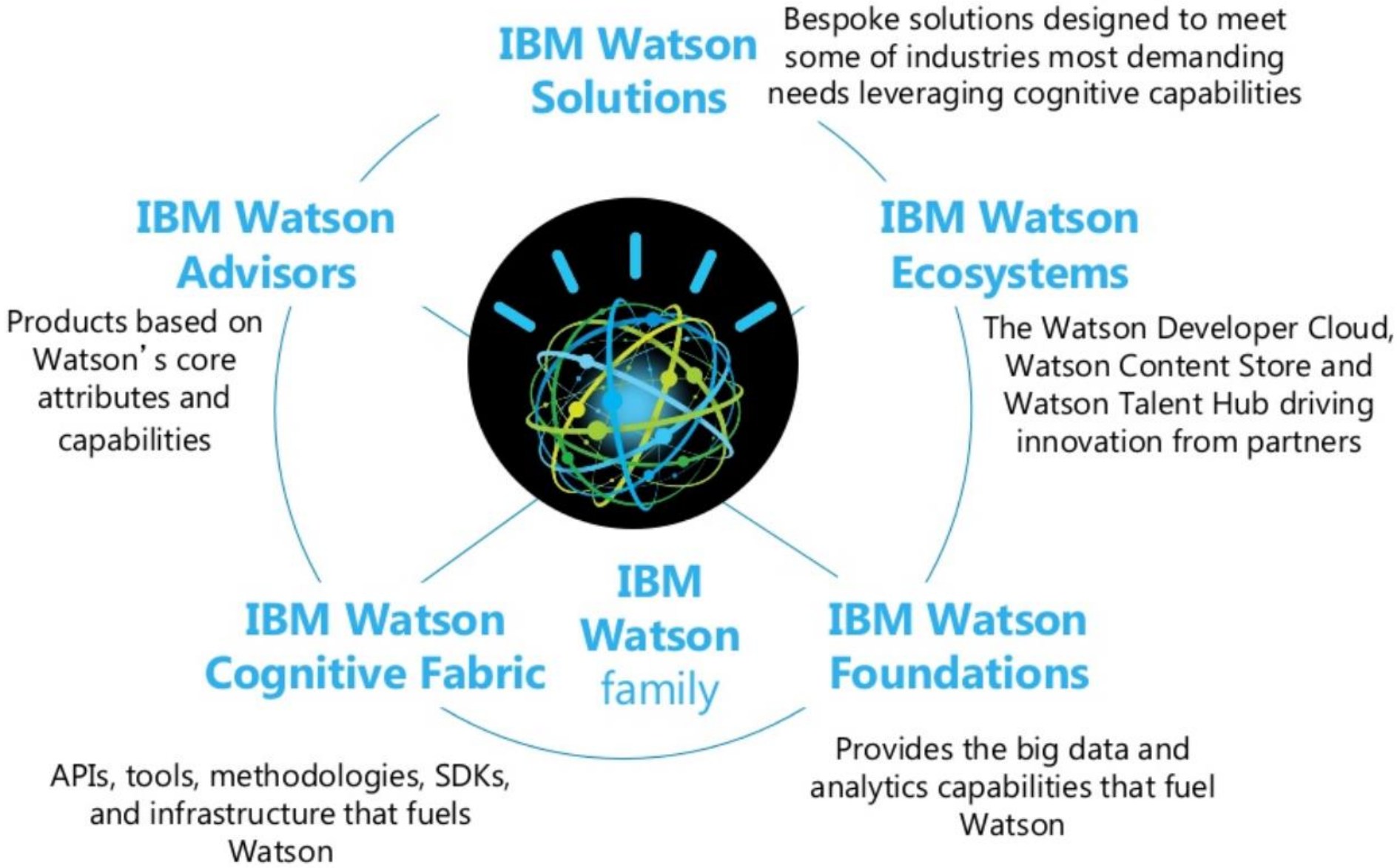
BigData on IBM Cloud

https://www.ibm.com/uk-en/it-infrastructure/solutions/big-data

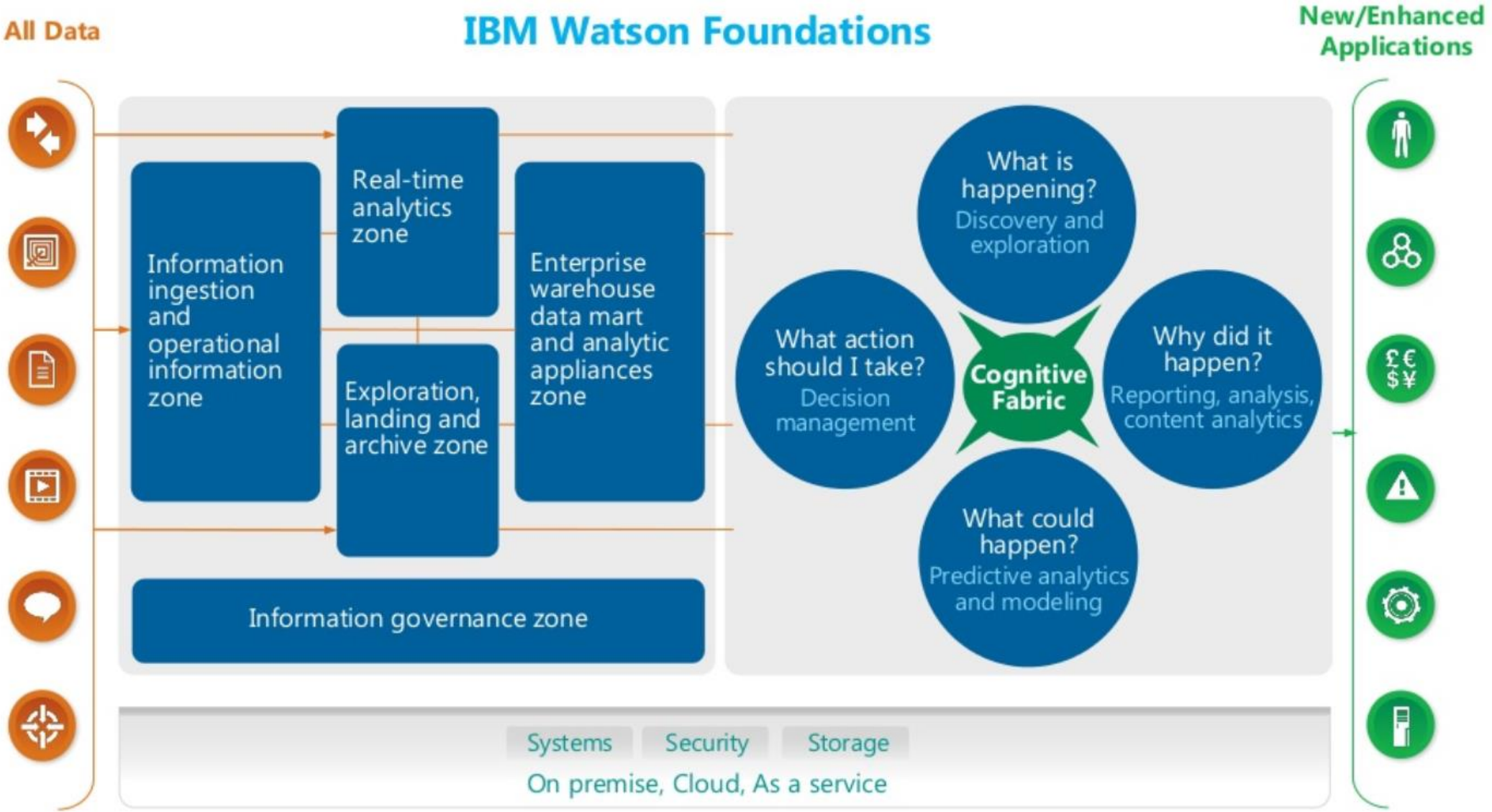
IBM Big Data Platform



IBM Watson Family



Big Data & Analytics Infrastructure



SCHOOL OF
HUMAN SCIENCES
& TECHNOLOGY