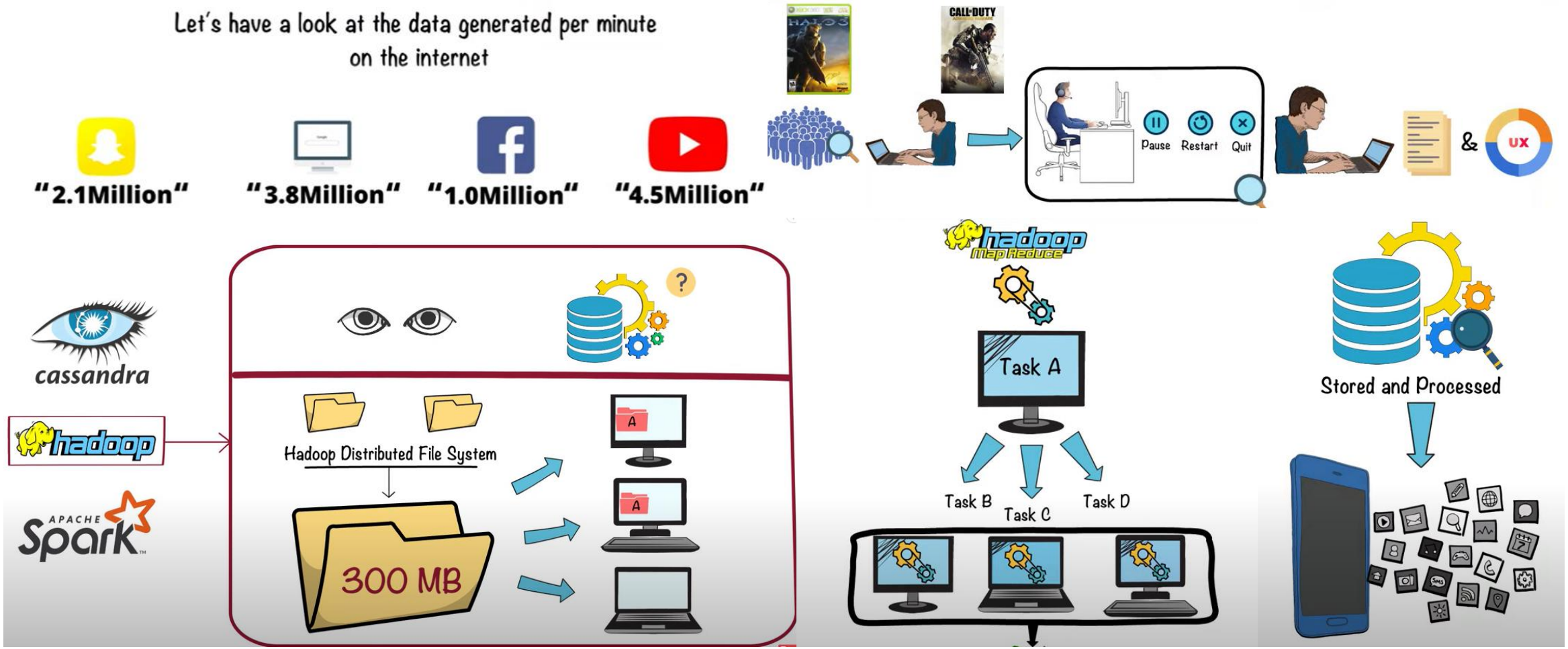
5. 빅데이터 플랫폼

박경규

빅데이터플랫폼



https://youtu.be/bAyrObl7TYE

데이터 형태

정형 데이터

- 형태가 있으며 연산 가능함. 주로 관계형 데이터베이스에 저장됨
- 데이터 수집 난이도가 낮음
- 내부 시스템인 경우가 대부분임
- 파일 형태의 스프레드시트라도 내부에 형식을 가지고 있어 처리가 쉬운 편임
- EX) 관계형 데이터베이스, 스프레드시트, CSV 등

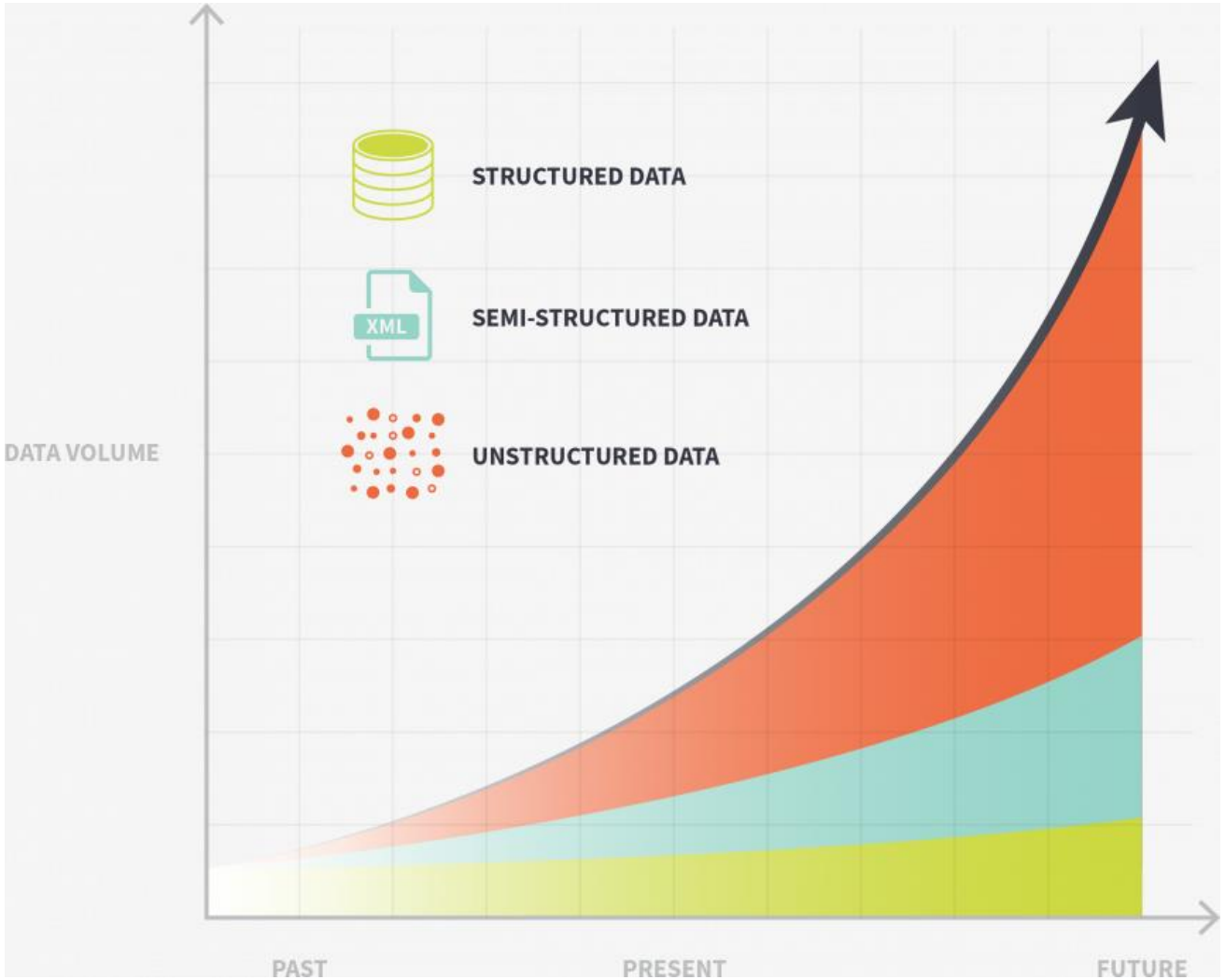
반정형 데이터

- 형태(스키마, 메타데이터)가 있으며 연산이 불가능
- 주로 파일에 저장됨
- 데이터 수집 난이도가 중간
- 보통 API 형태로 제공되기 때문에 데이터 처리기술이 요구됨
- EX) XML, HTML, 로그형태(웹로그, 센서데이터), Machine Data 등

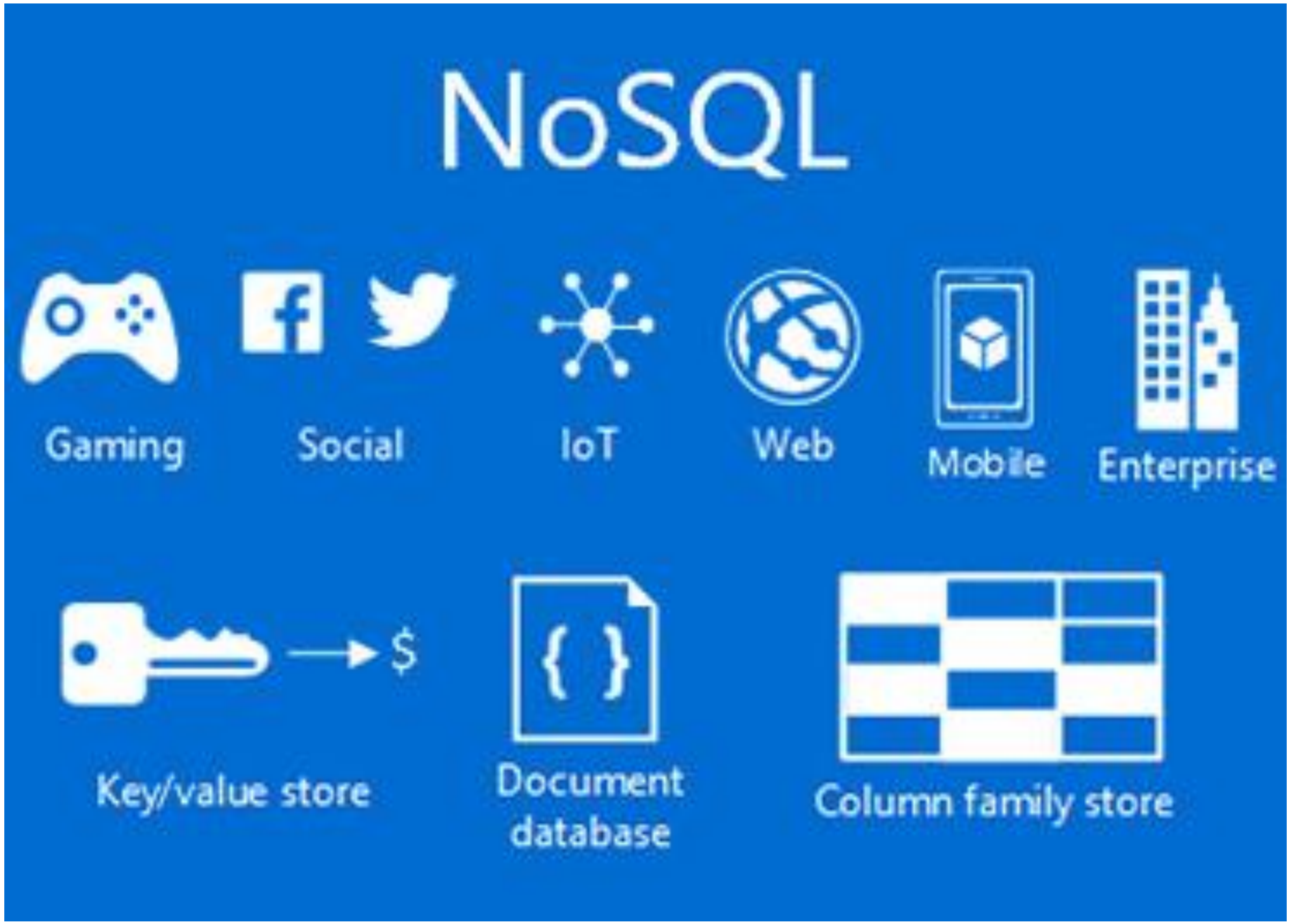
비정형 데이터

- 형태가 없으며, 연산이 불가능함
- 주로 NoSQL에 저장됨
- 데이터 수집 난이도가 높음
- 텍스트 마이닝 혹은 파일일 경우 파일을 데이터 형태로 파싱해야 하기 때문에 수집 데이터 처리가 어려움
- EX) 소셜데이터(트위터, 페이스북), 이메일, 보고서

비정형 데이터


STRUCTURED DATA
XML
SEMI-STRUCTURED DATA
UNSTRUCTURED DATA
DATA VOLUME
PAST
PRESENT
FUTURE

NoSQL



mongoDB

amazon DynamoDB

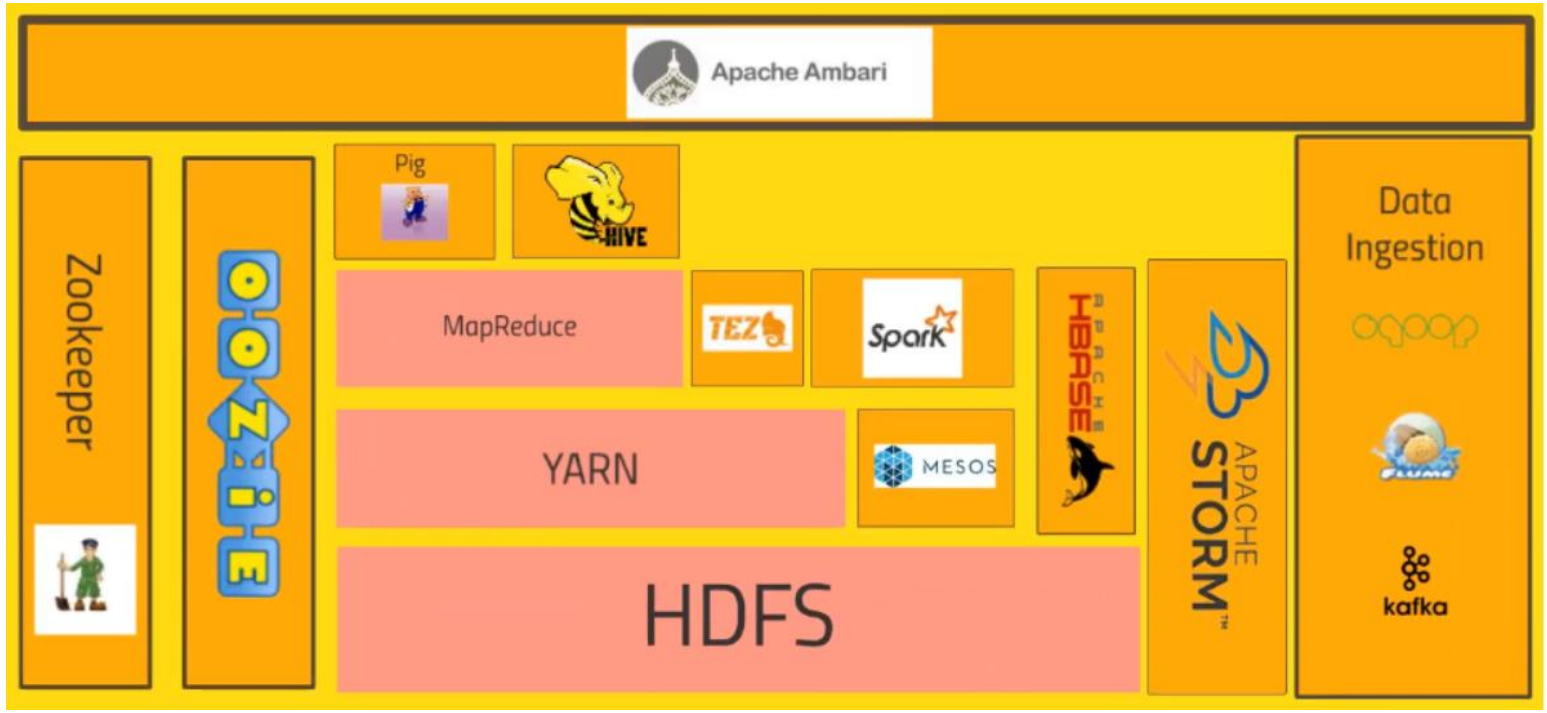
Elasticsearch

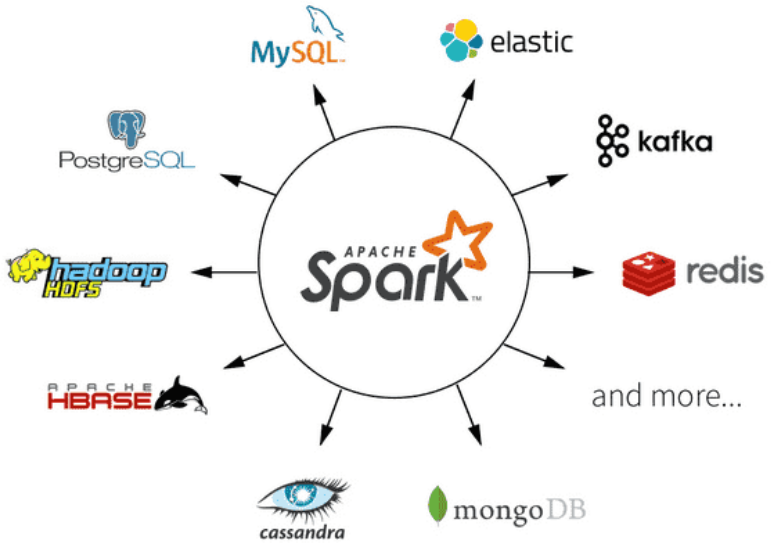
cassandra

redis

APACHE HBASE

빅데이터 프로세싱 솔루션





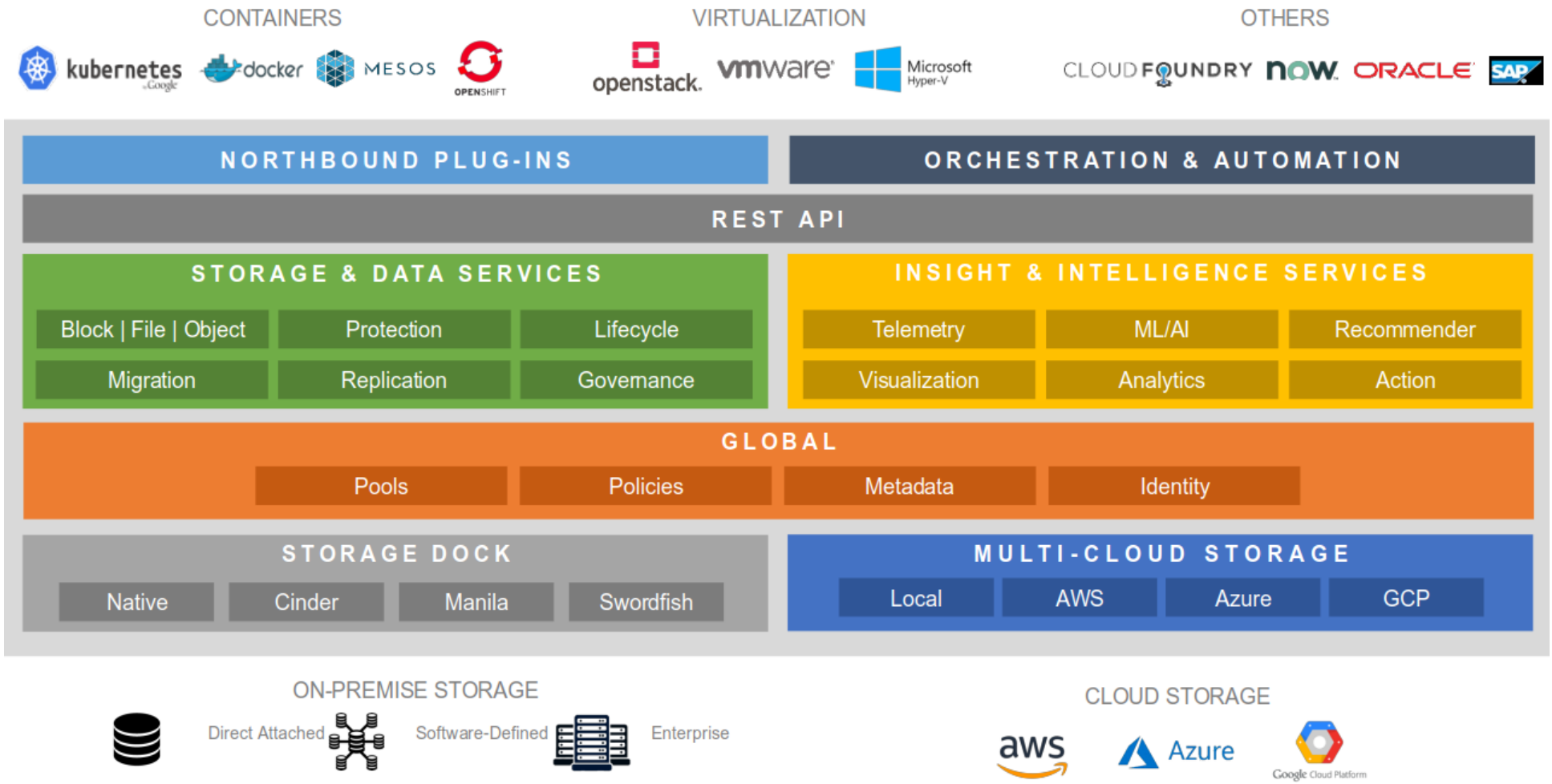
Data Collection

Data Aggregation & Processing

Indexing & storage

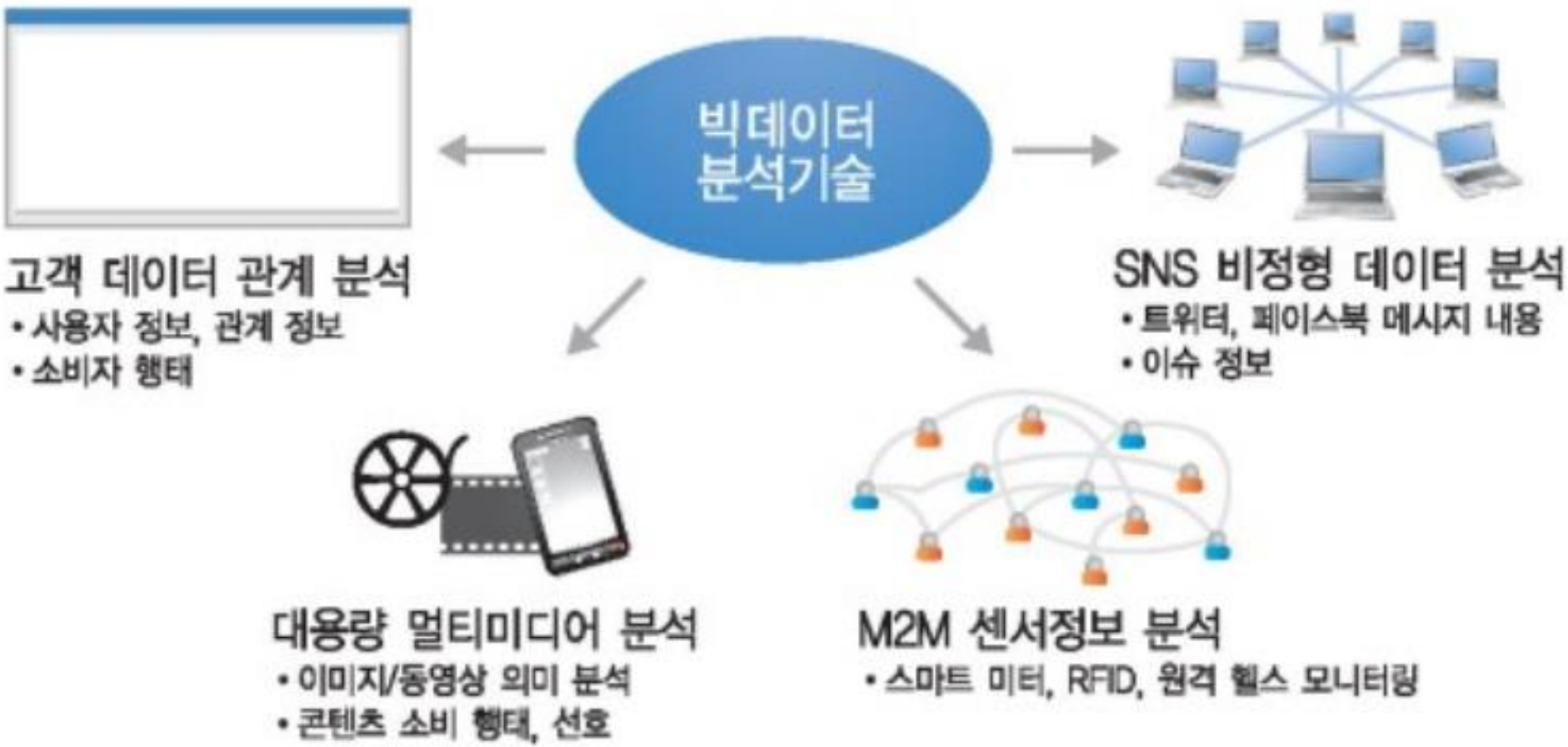
Analysis & visualization

데이터플랫폼 아키텍처



https://www.opensds.io/about/

빅데이터 분석



https://bitnine.tistory.com/393

빅데이터 기술

■ 빅데이터 저장기술

- 다양하고 많은 양의 빅데이터를 저장하고 관리하는 기술이 필수
- 대표적인 빅데이터 저장기술 : 하둡(Hadoop), NoSQL(Not Only SQL)
- 하둡(Hadoop) : 대용량 데이터를 분산 처리할 수 있는 자바 기반의 오픈 소스 프레임워크
- NoSQL(Not Only SQL) : 관계형데이터베이스의 일관성 특징보다는 가용성과 확장성에 중점을 둔 데이터베이스

■ 빅데이터 분석기술

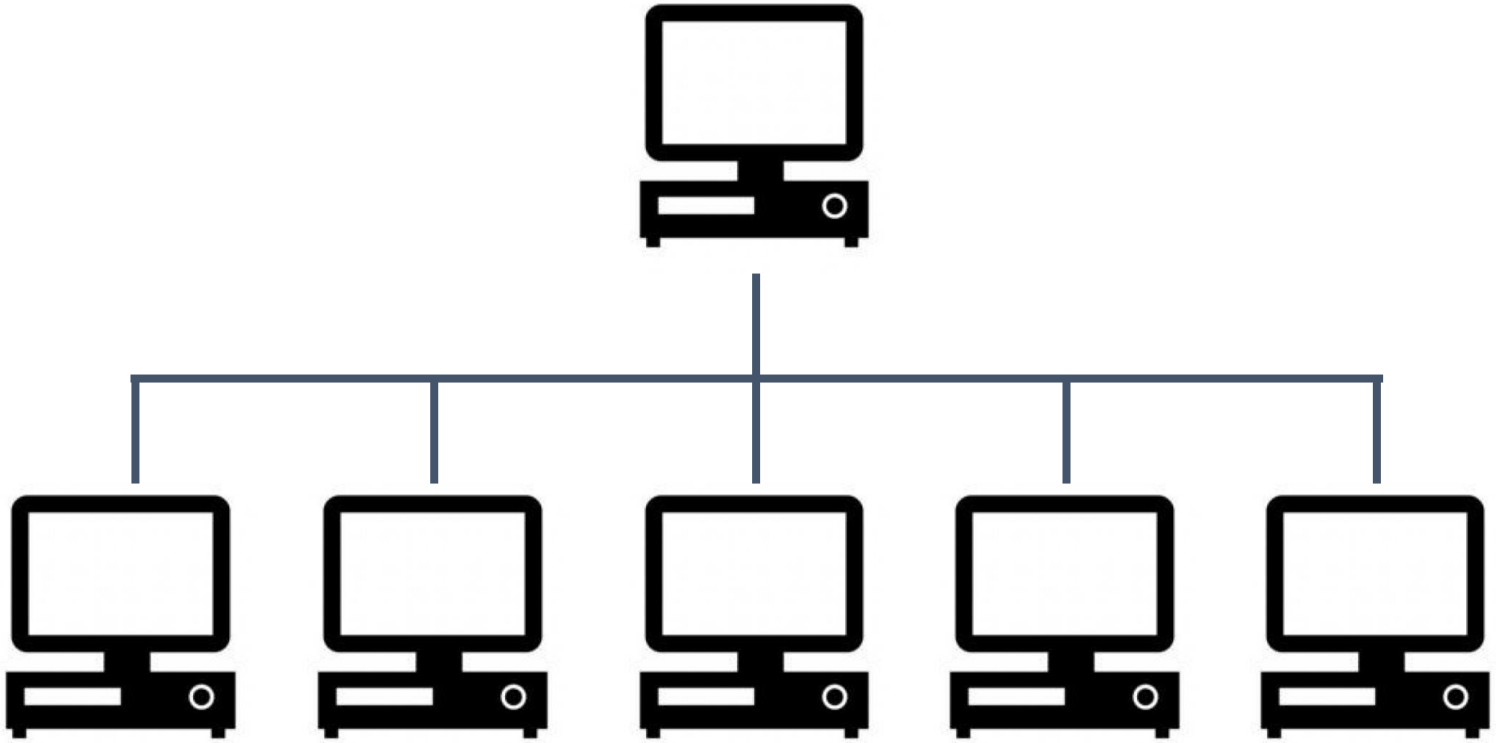
- 텍스트 마이닝, 오피니언 마이닝, 소셜 네트워크 분석, 패턴인식, 머신러닝, 딥러닝, 자연어 처리 기술 활용
- 텍스트 마이닝(text mining) : 텍스트 데이터에서 자연어 처리 기술을 기반해 가치 있는 정보를 추출하고 가공
- 오피니언 마이닝(opinion mining) : SNS, 블로그 게시글 등에서 사용자들의 의견을 수집하여 제품과 서비스에 대한 감성을 파악하거나 유용한 정보로 재가공하는 기술
- 소셜 네트워크 분석(social network analysis) : 소셜 네트워크상에서의 영향력인 사람/데이터 등 객체 간의 관계/특성을 분석하고 시각화하는 기법

분산시스템

- 로컬 컴퓨터에서는 RAM 용량(예: 32GB) 범위의 데이터를 다룰 수 있습니다.
- 더 큰 데이터셋은 SQL 데이터베이스를 사용하여 스토리지를 사용하거나,
  여러대의 컴퓨터로 구성된 분산시스템을 사용해야 합니다.

Local

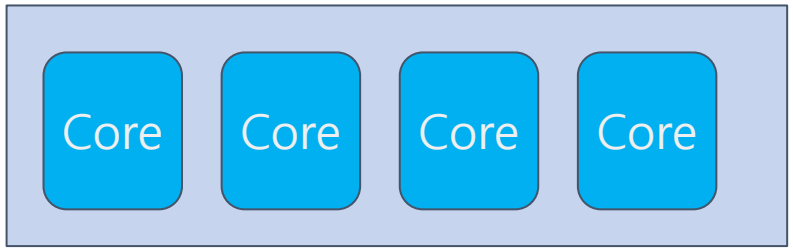
Distributed



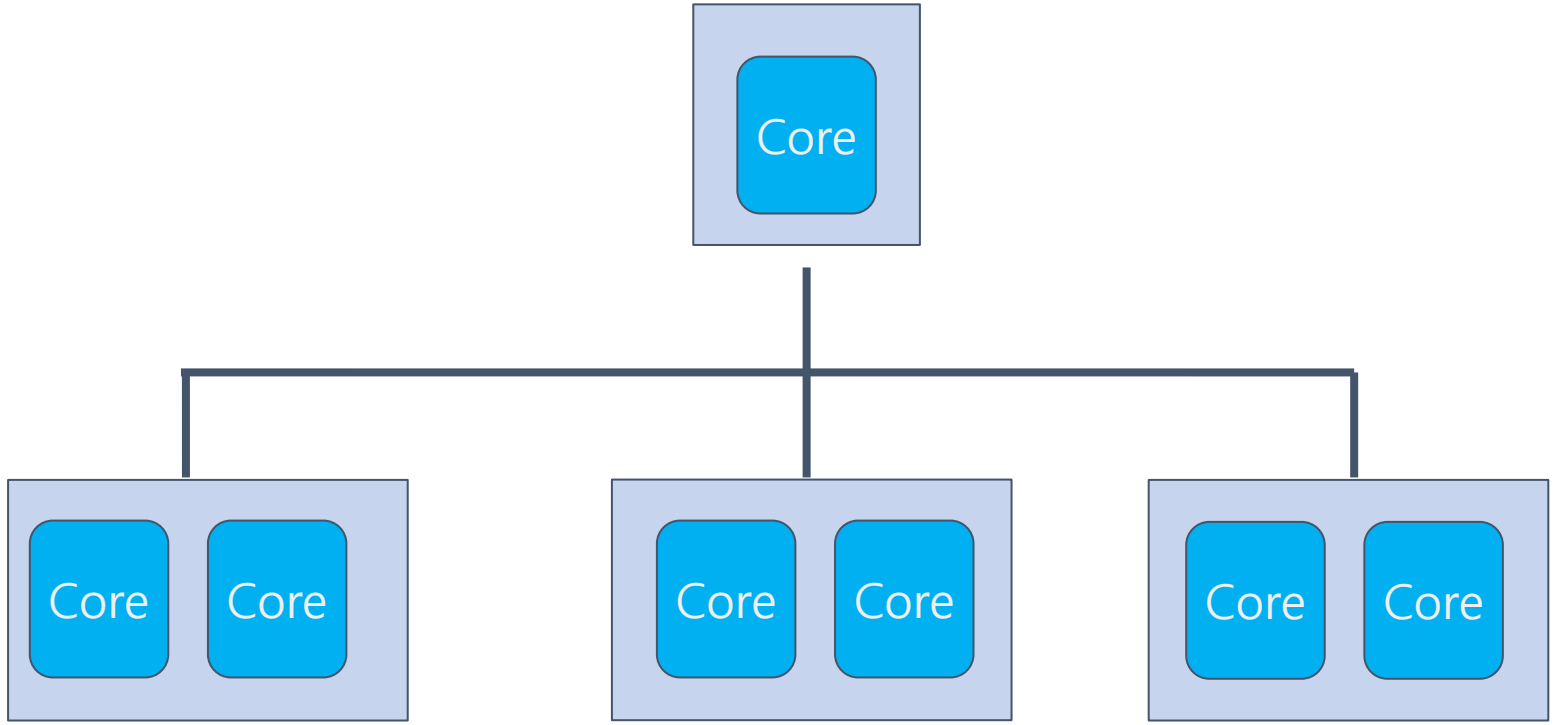
분산시스템

- 로컬 프로세스는 단일 시스템의 컴퓨팅 리소스를 사용합니다.
- 분산 프로세스는 네트워크를 통해 연결된 여러 머신의 컴퓨팅 리소스를 액세스 할 수 있습니다.
- 단일 머신을 Scale Up 하는 것보다 여러대의 머신으로 확장(Scale Out)하는 것이 더 쉽습니다.
- 분산시스템에서는 한 대의 시스템에 장애가 발생해도 전체 네트워크가 계속 작동 할 수 있습니다.

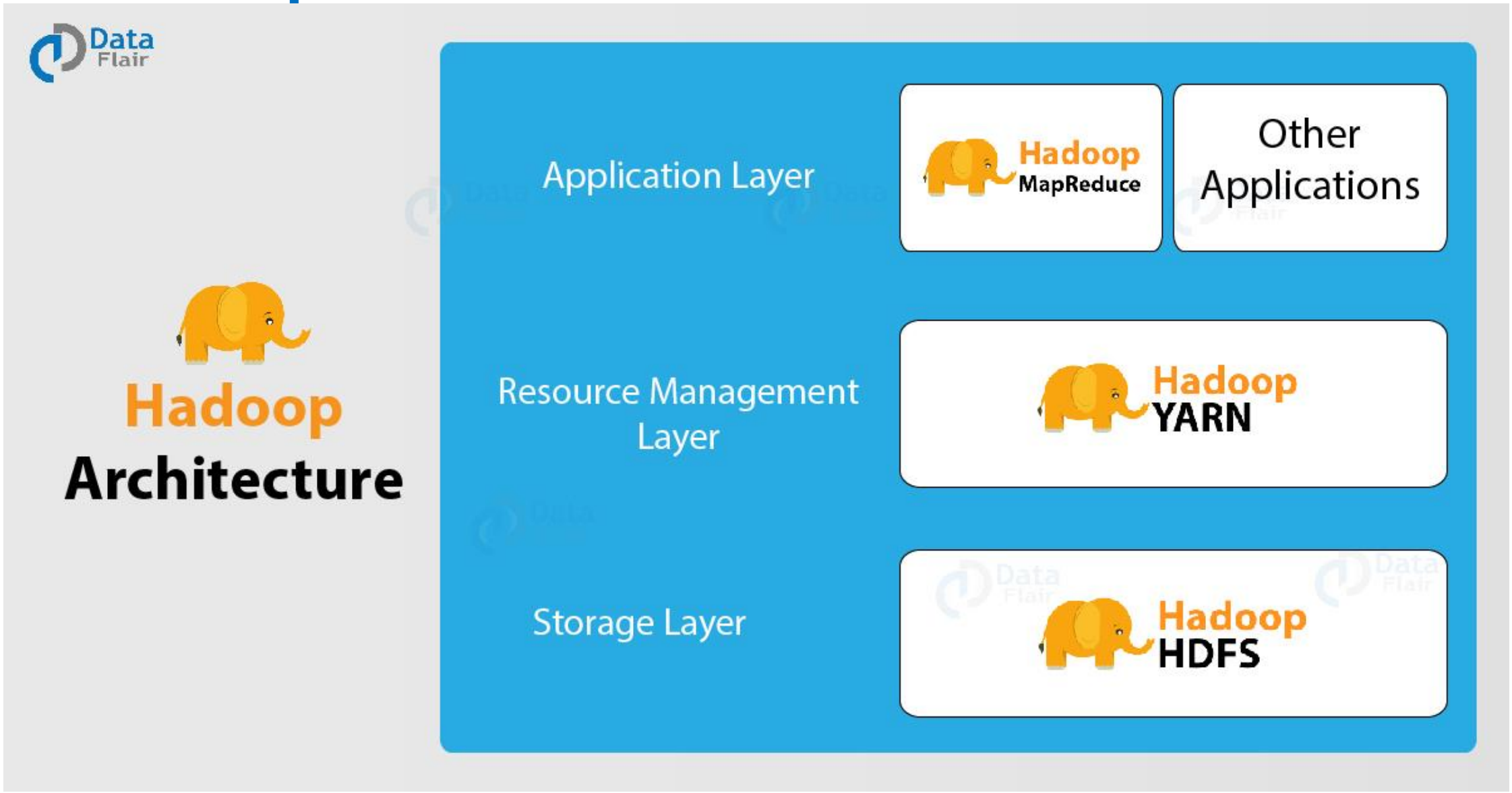
Local



Distributed



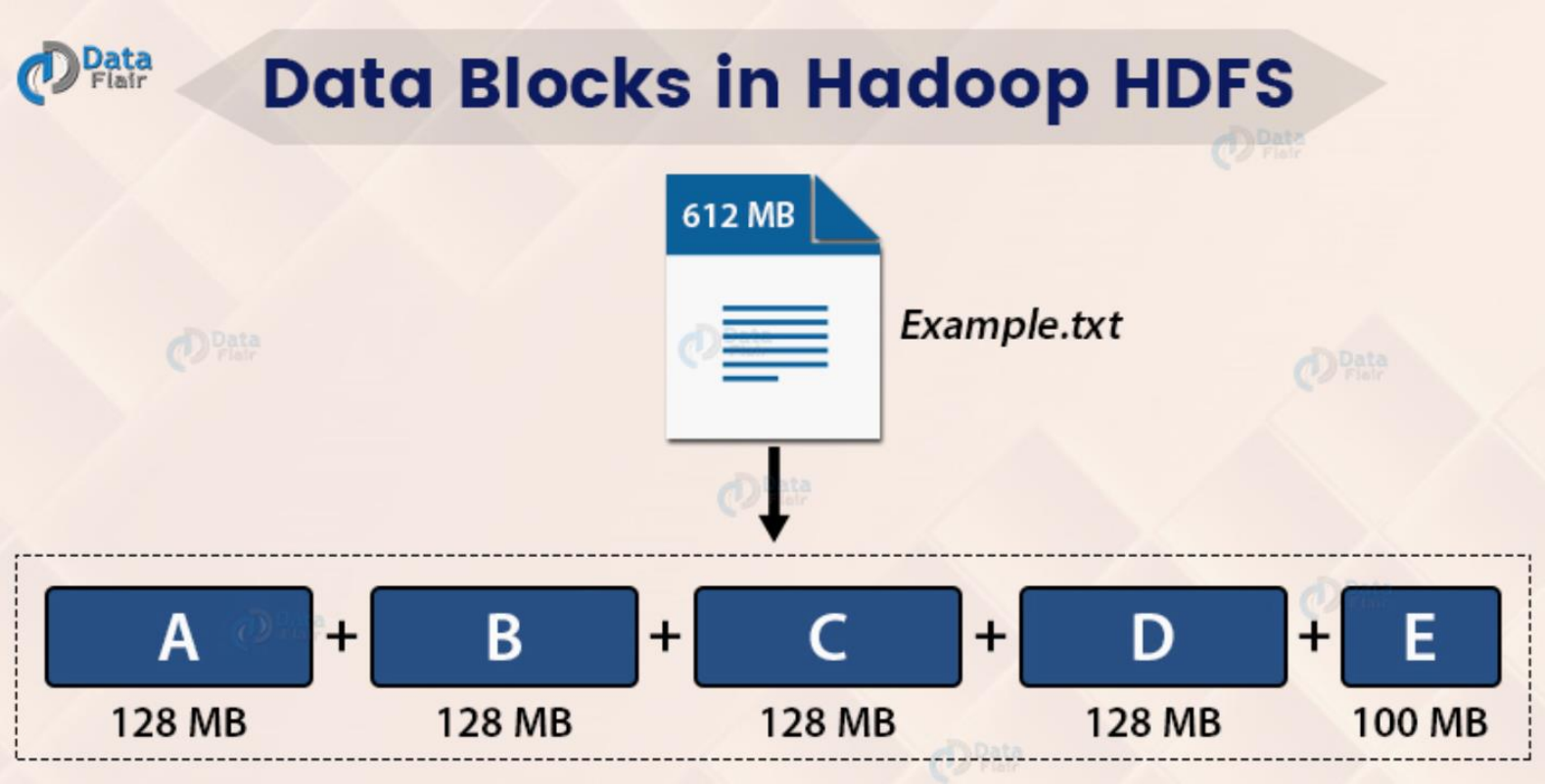
하둡(Hadoop)



https://data-flair.training/blogs/hadoop-architecture/

하둡 HDFS 파일 저장 방식

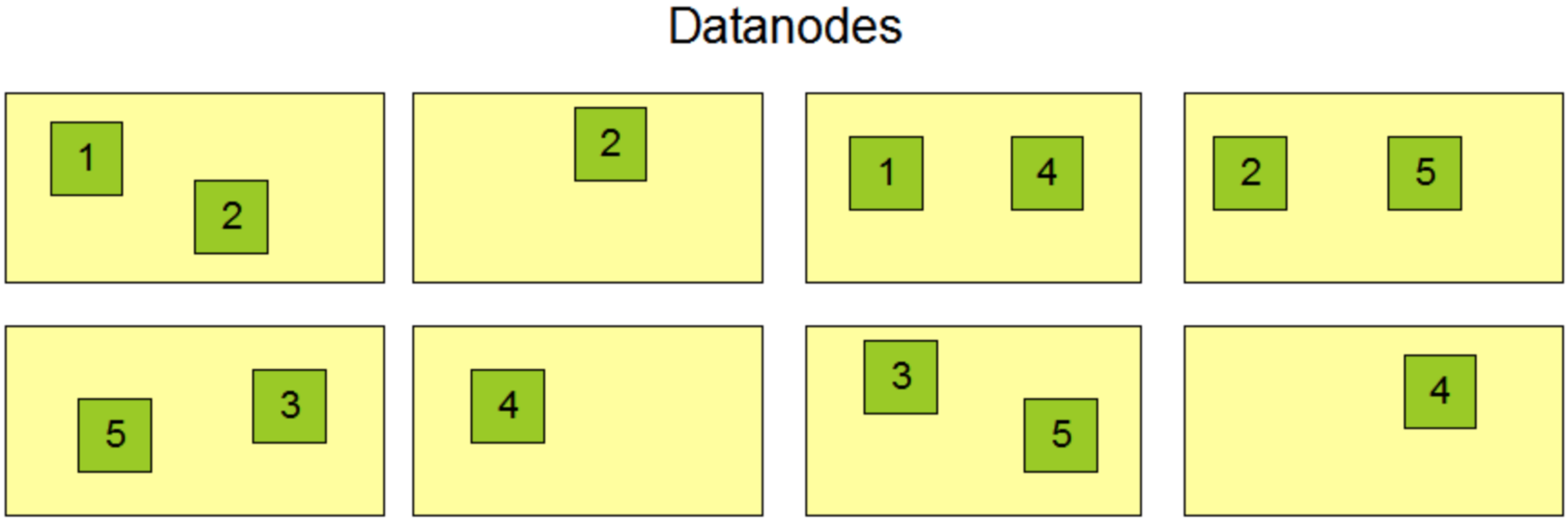
- HDFS는 파일을 분산 저장하기 위해서 먼저 파일의 메타 데이터와 컨텐츠 데이터를 분리합니다.
- 메타 데이터 : 파일의 접근 권한, 생성일, 수정일, 네임 스페이스 등 파일에 대해 설명하는 정보
- 컨텐츠 데이터 : 실제 파일에 저장된 데이터
- 파일의 메타데이터는 네임 노드에 저장되며 컨텐츠 데이터는 블록 단위로 쪼개져서 데이터 노드에 저장 됩니다.



https://yeomko.tistory.com/38

하둡 HDFS 데이터 노드

- 파일의 컨텐츠 데이터는 블록 단위로 나뉘며, 블록의 기본 크기는 128MB이며 최소 3개의 복사본을 생성하여 분산 저장합니다.
- 데이터 노드는 그 중 하나의 복사본을 저장하는 것입니다.



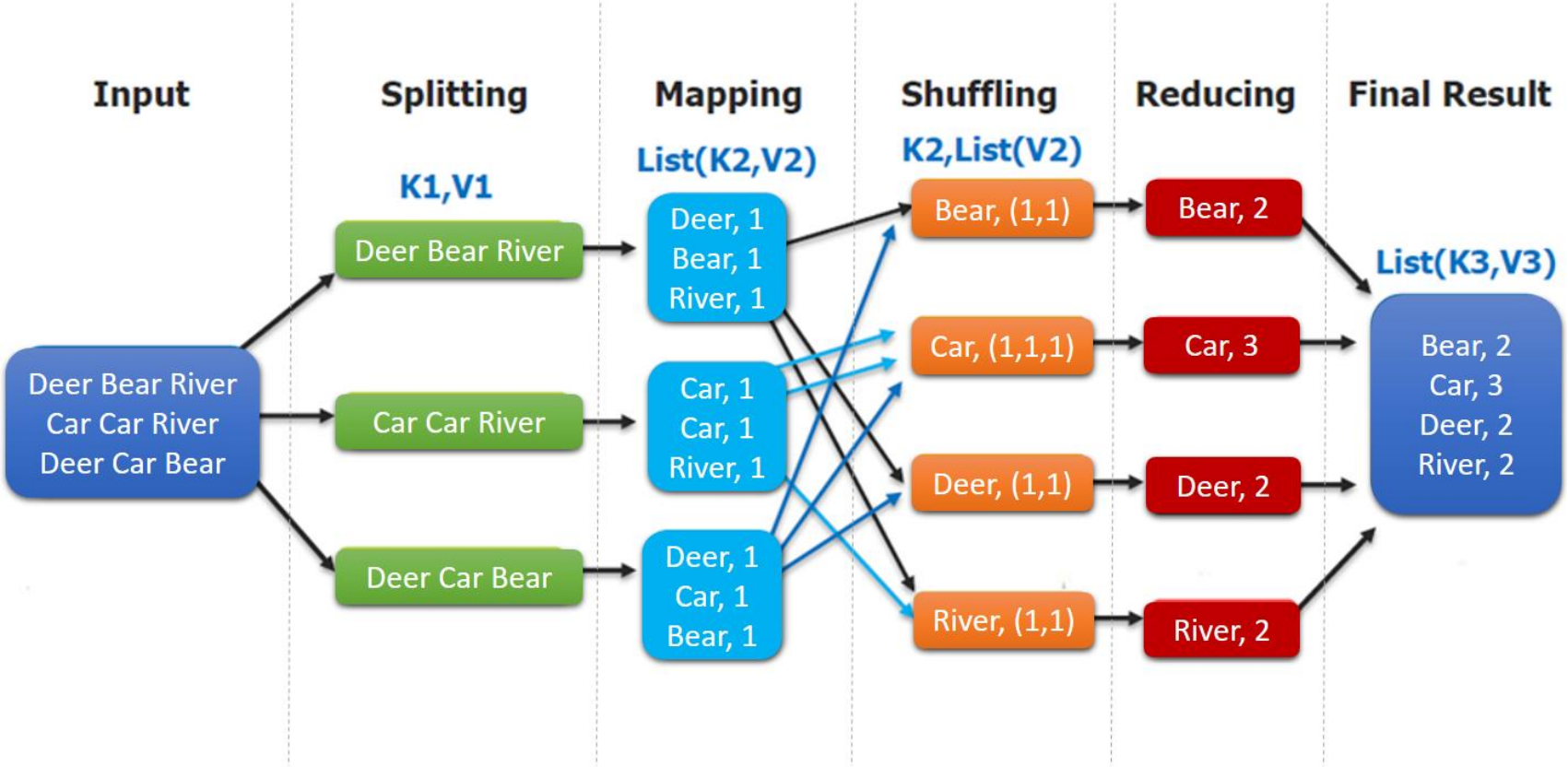
https://yeomko.tistory.com/38

하둡 맵리듀스(MapReduce)

- 맵리듀스는 구글 내부에서 크롤링 된 문서, 로그 등 방대한 양의 raw data를 분석하는 과정에서 느낀 불편함에서 출발했습니다.
- 프로그램 로직 자체는 단순한데 입력 데이터의 크기가 워낙 커서 연산을 하나의 물리 머신에서 수행할 수가 없었습니다.
- 거대한 인풋 데이터를 쪼개어 수많은 머신들에게 분산시켜서 로직을 수행한 다음 결과를 하나로 합치자는 것이 핵심 아이디어 입니다.
- MapReduce 프레임워크에서 개발자가 코드를 작성하는 부분은 map과 reduce 두 가지 함수입니다.
- map은 전체 데이터를 쪼갠 청크에 대해서 실제로 수행할 로직입니다.
- reduce는 분산되어 처리된 결과 값들을 다시 하나로 합쳐주는 과정이며, 이 역시 분산된 머신들에서 병렬적으로 수행됩니다.

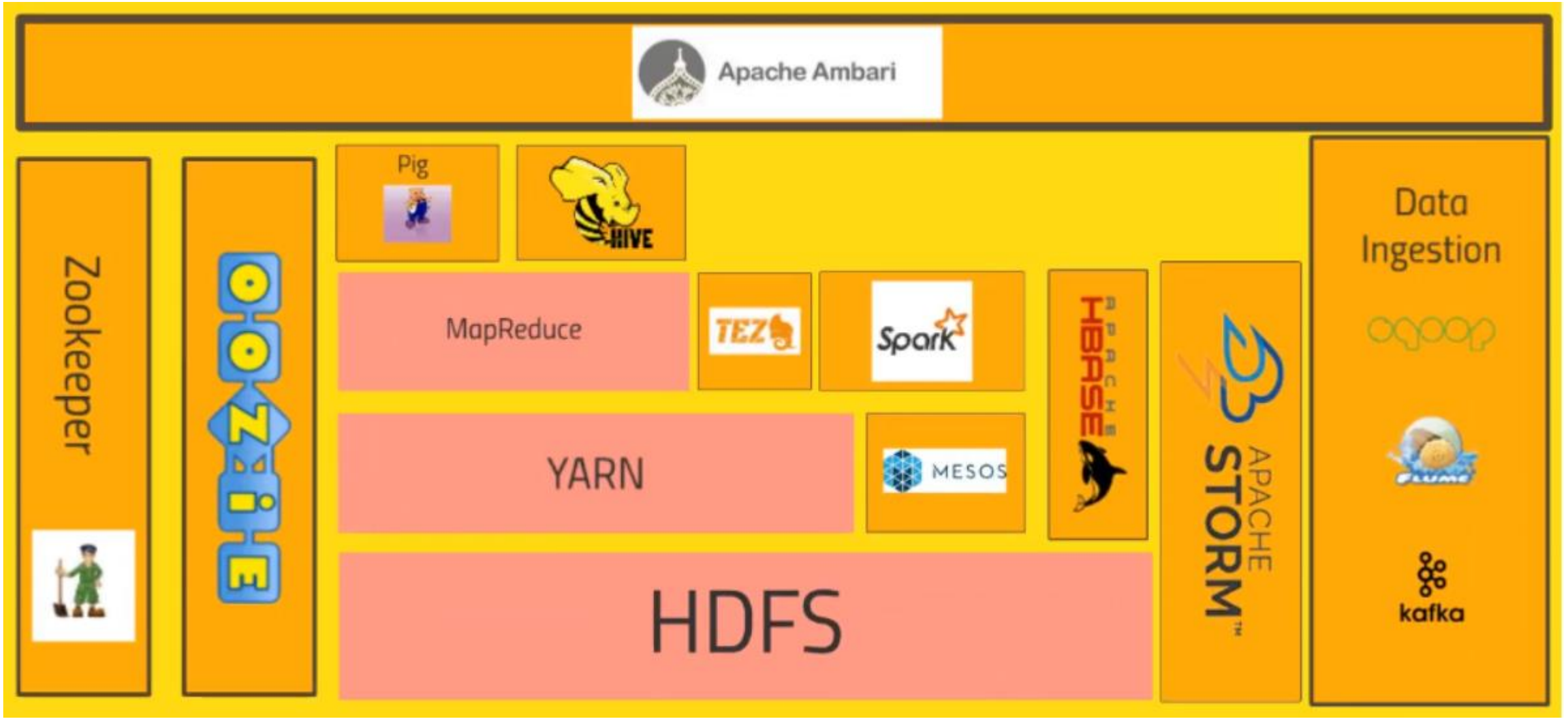
하둡 맵리듀스 수행 절차

1. 쪼개기(Split): 크기가 큰 인풋파일을 작은 단위의 청크들로 나누어 분산 파일 시스템에 저장합니다.
2. 데이터 처리하기(Map): 잘게 쪼개어진 파일을 인풋으로 받아서 데이터를 분석하는 로직을 수행합니다.
3. 처리된 데이터 합치기(Reduce): 처리된 데이터를 다시 합칩니다.



MapReduce Word Count Process

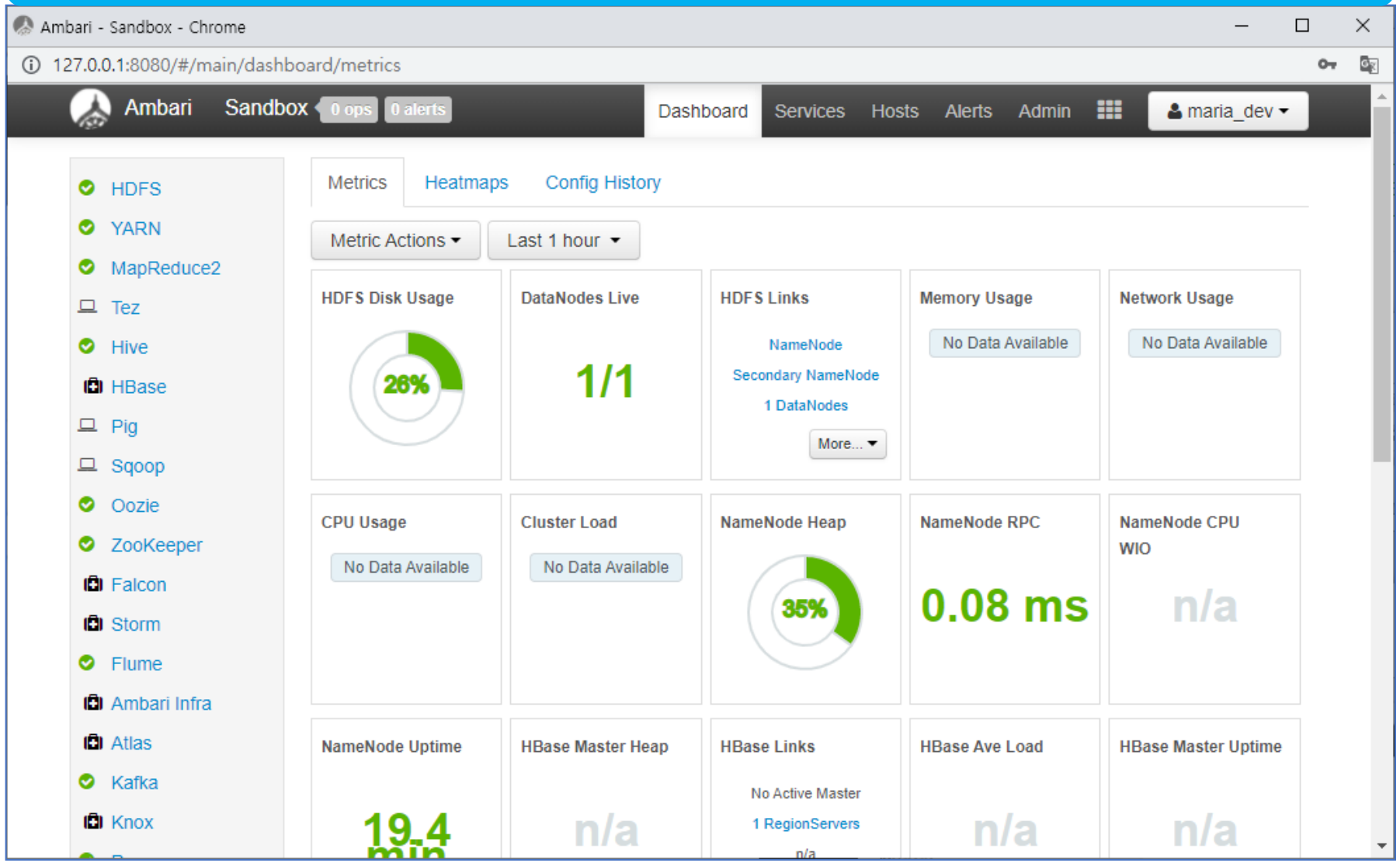
하둡 에코시스템



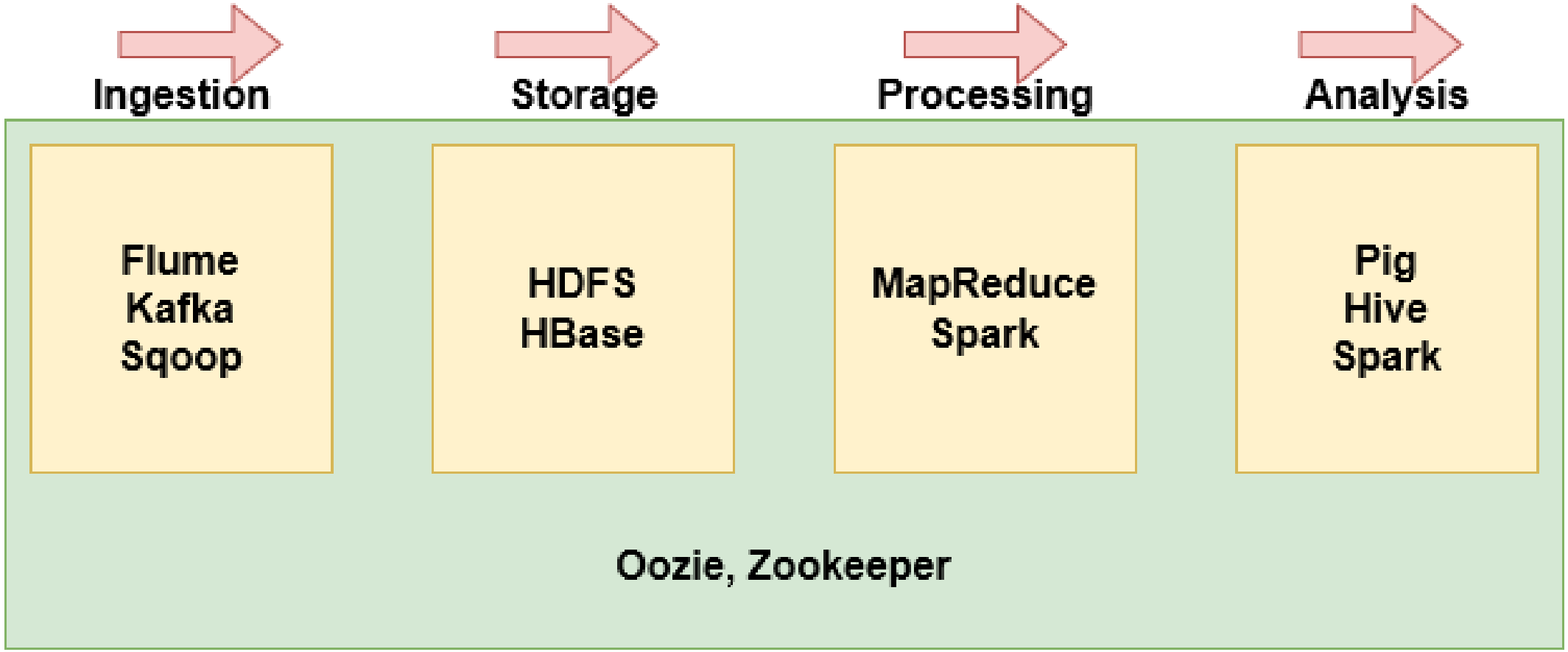
https://csschoolonline.xyz/hadoop-ecosystem-introduction-to-big-data-data-engineering/

HDP(Hortonworks Data Platform)

https://www.cloudera.com/downloads/hortonworks-sandbox.html



빅데이터 프로세싱



https://www.analyticsvidhya.com/blog/2020/10/introduction-hadoop-ecosystem/

빅데이터 분산 프로세싱 엔진 스파크(Spark)

A fast and general engine for large-scale data processing

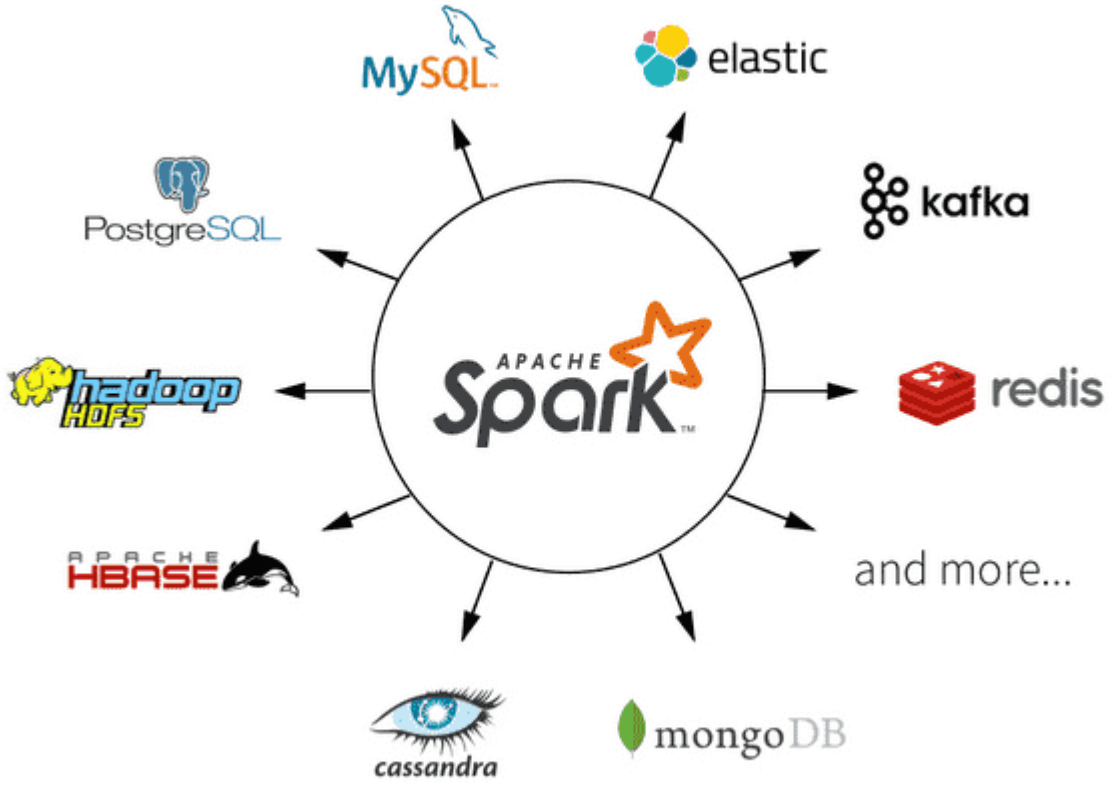
It's scalable

It's fast

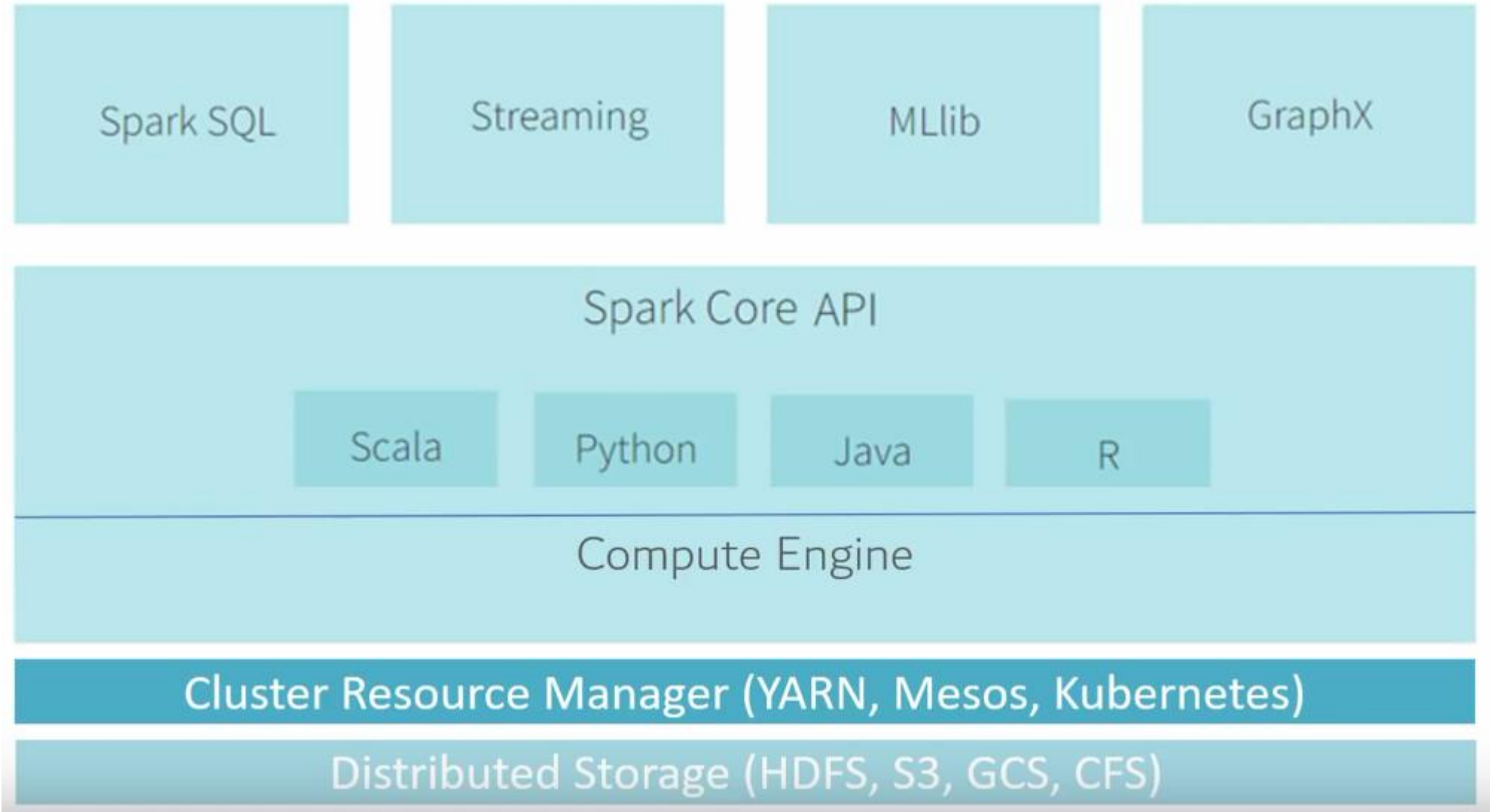
It's hot

It's not that hard

Code in Python, Java, or Scala



스파크의 구조


Spark SQL
Streaming
MLlib
GraphX
Spark Core API
Scala
Python
Java
R
Compute Engine
Cluster Resource Manager (YARN, Mesos, Kubernetes)
Distributed Storage (HDFS, S3, GCS, CFS)

Colab에서 PySpark 사용하는 방법

spark_in_colab.ipynb

spark_in_colab.ipynb - Colabora

colab.research.google.com/drive/1mDiTzbBhcORCs7OcG7XVKZMIHxp2Y0EA#scrollTo=baECVey4xrv8

spark_in_colab.ipynb

파일 수정 보기 삽입 런타임 도구 도움말 모든 변경사항이 저장됨

댓글 공유

+ 코드 + 텍스트 연결 수정 가능

Java Virtual Machine (JVM) 설치

```
[ ]  !apt-get install openjdk-8-jdk-headless
```

Apache Spark 설치

```
[ ]  !wget -q https://downloads.apache.org/spark/spark-3.1.2/spark-3.1.2-bin-hadoop2.7.tgz
```

```
[ ]  !tar xf spark-3.1.2-bin-hadoop2.7.tgz
```

```
[ ]  !ls -lt
```

Colab에서 PySpark 사용하는 방법

findspark 라이브러리 설치

```
[ ]  !pip install -q findspark
```

환경변수 설정

```
[ ]  import os
     os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64"
     os.environ["SPARK_HOME"] = "/content/spark-3.1.2-bin-hadoop2.7"
```

테스트

```
[ ]  import findspark
     findspark.init()
```

```
[ ]  findspark.find()
```

```
'/content/spark-3.1.2-bin-hadoop2.7'
```

PySpark DataFrame 실습

spark_dataframe.ipynb

■ 데이터 파일 : people.json

{"name":"Michael"}

{"name":"Andy", "age":30}

{"name":"Justin", "age":19}

■ Creating a DataFrame

```
[2]  from pyspark.sql import SparkSession
```

```
[3]  # May take a little while on a local computer
     spark = SparkSession.builder.appName("Basics").getOrCreate()
```

```
[4]  df = spark.read.json('people.json')
```

PySpark DataFrame 실습

■ Showing the data

```
[5]  # Note how data is missing!
     df.show()
```

```
+----+-------+
| age|   name|
+----+-------+
|null|Michael|
|  30|   Andy|
|  19| Justin|
+----+-------+
```

```
[6]  df.printSchema()
```

```
root
 |-- age: long (nullable = true)
 |-- name: string (nullable = true)
```

```
[7]  df.columns
```

```
['age', 'name']
```

```
[8]  df.describe()
```

```
DataFrame[summary: string, age: string, name: string]
```

PySpark DataFrame 실습

■ Infer schema

```
[9]  from pyspark.sql.types import StructField,StringType,IntegerType,StructType
```

```
[10] data_schema = [StructField("age", IntegerType(), True),StructField("name", StringType(), True)]
```

```
[11] final_struc = StructType(fields=data_schema)
```

```
[12] df = spark.read.json('people.json', schema=final_struc)
```

```
[13] df.printSchema()
```

```
root
 |-- age: integer (nullable = true)
 |-- name: string (nullable = true)
```

PySpark DataFrame 실습

■ Grabbing the data

```
[14]  df['age']
```

```
Column<'age'>
```

```
[15]  type(df['age'])
```

```
pyspark.sql.column.Column
```

```
[16]  df.select('age')
```

```
DataFrame[age: int]
```

```
[17]  type(df.select('age'))
```

```
pyspark.sql.dataframe.DataFrame
```

```
[18]  df.select('age').show()
```

```
+----+
| age|
+----+
|null|
|  30|
|  19|
+----+
```

PySpark DataFrame 실습

■ Multiple Columns

```
[20]  df.select(['age','name'])
```

```
DataFrame[age: int, name: string]
```

```
[21]  df.select(['age','name']).show()
```

```
+----+-------+
| age|   name|
+----+-------+
|null|Michael|
|  30|   Andy|
|  19| Justin|
+----+-------+
```

■ Creating new columns

```
[22]  # Adding a new column with a simple copy
      df.withColumn('newage',df['age']).show()
```

```
+----+-------+------+
| age|   name|newage|
+----+-------+------+
|null|Michael|  null|
|  30|   Andy|    30|
|  19| Justin|    19|
+----+-------+------+
```

PySpark DataFrame 실습

■ More complicated operations to create new columns

```
[25]  df.withColumn('doubleage',df['age']*2).show()
```

```
+----+-------+---------+
| age|   name|doubleage|
+----+-------+---------+
|null|Michael|     null|
|  30|   Andy|       60|
|  19| Justin|       38|
+----+-------+---------+
```

```
[26]  df.withColumn('add_one_age',df['age']+1).show()
```

```
+----+-------+-----------+
| age|   name|add_one_age|
+----+-------+-----------+
|null|Michael|       null|
|  30|   Andy|         31|
|  19| Justin|         20|
+----+-------+-----------+
```

■ PySpark 튜토리얼

https://sparkbyexamples.com/pyspark-tutorial/

Thank you

kgpark88@gmail.com