Intro to Docker, Hadoop and Hive

leifsyliongka@gmail.com

Big Data: A Reintroduction

Reductionism

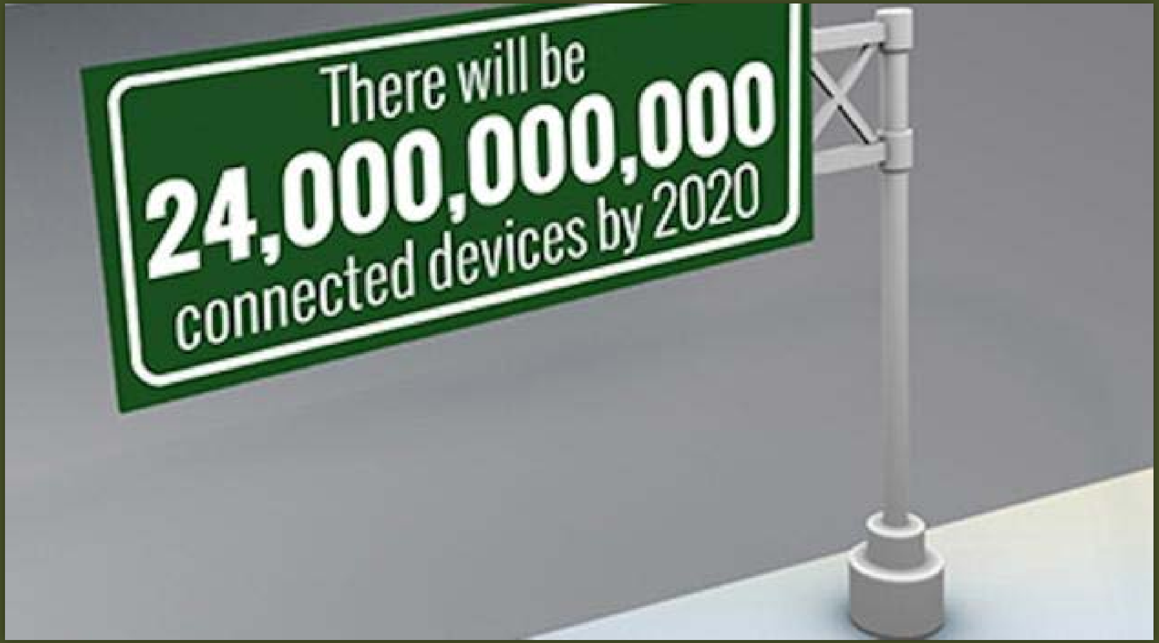
- Looking at the components of a system to understand the complex system
- Examples
    - An Ant
    - A neuron
    - An Internet user

Complexity

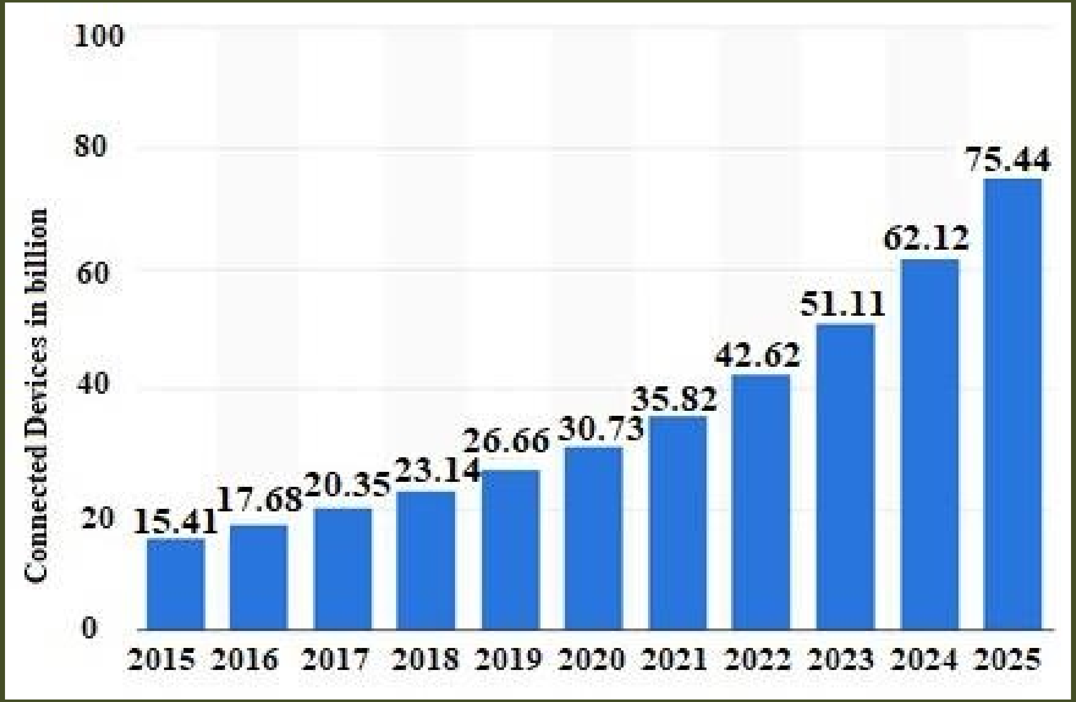
- Looking at a system as a whole rather than diving it to its very parts

- Examples
  - Ant Colony
  - Brain
  - Social Media Users

Prediction: “Internet of Things”

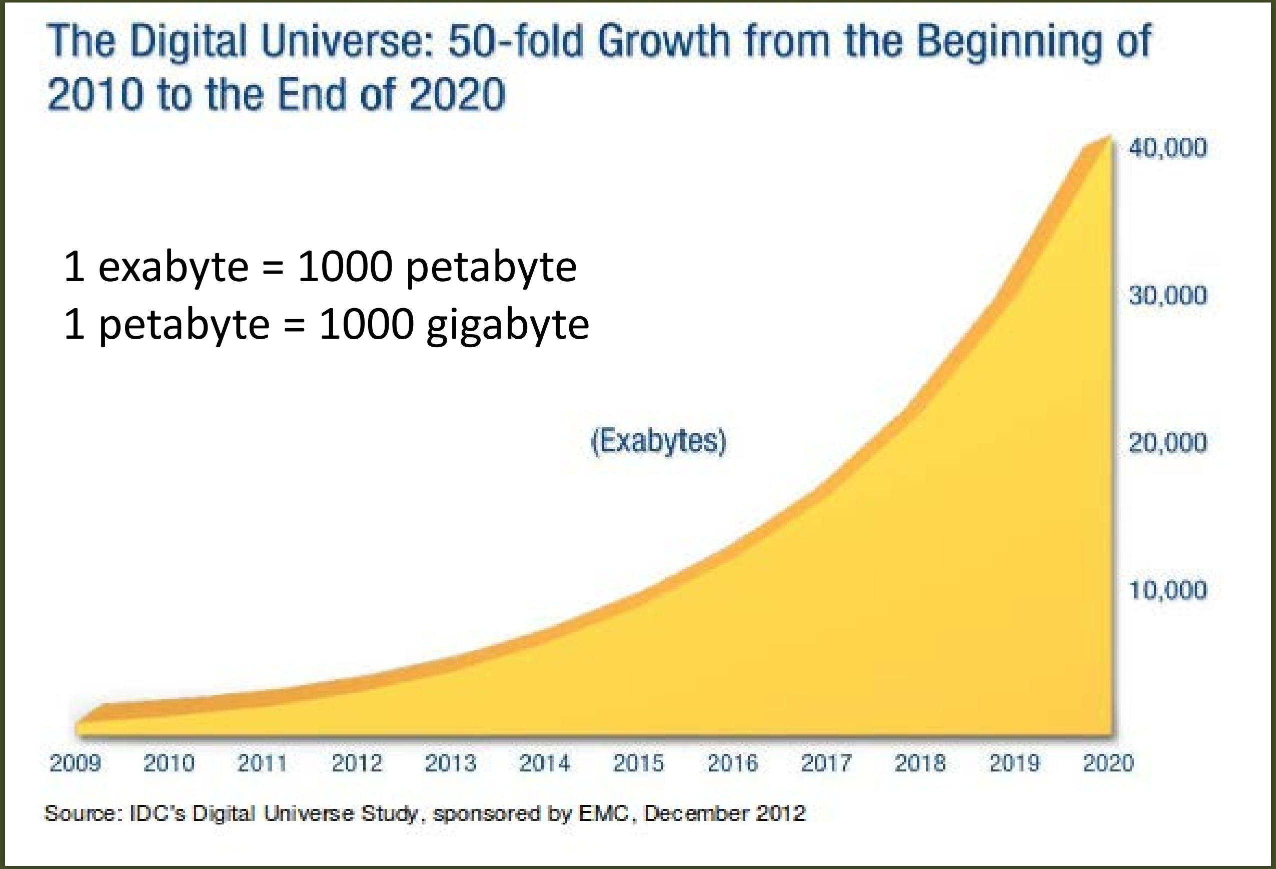


Prediction from 2013

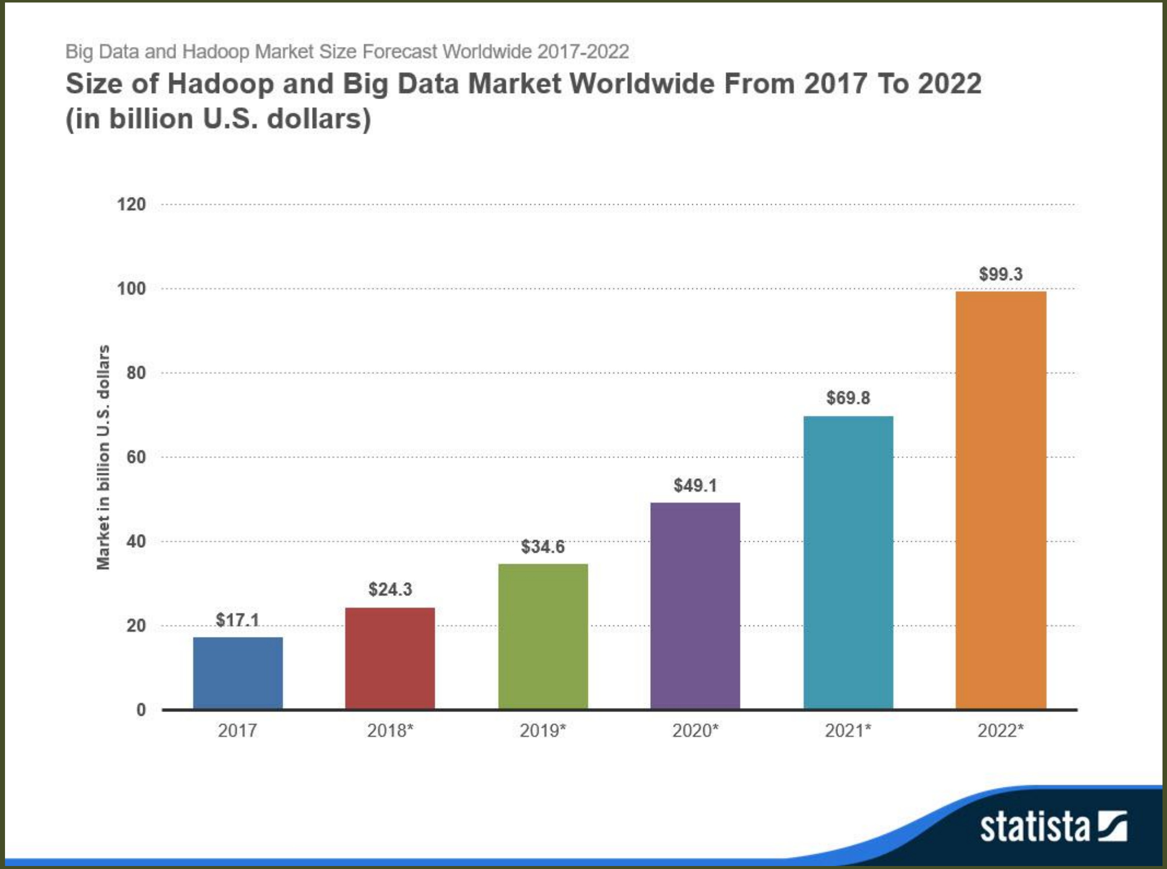


Prediction from 2018

Data: Growth



1 exabyte = 1000 petabyte
1 petabyte = 1000 gigabyte

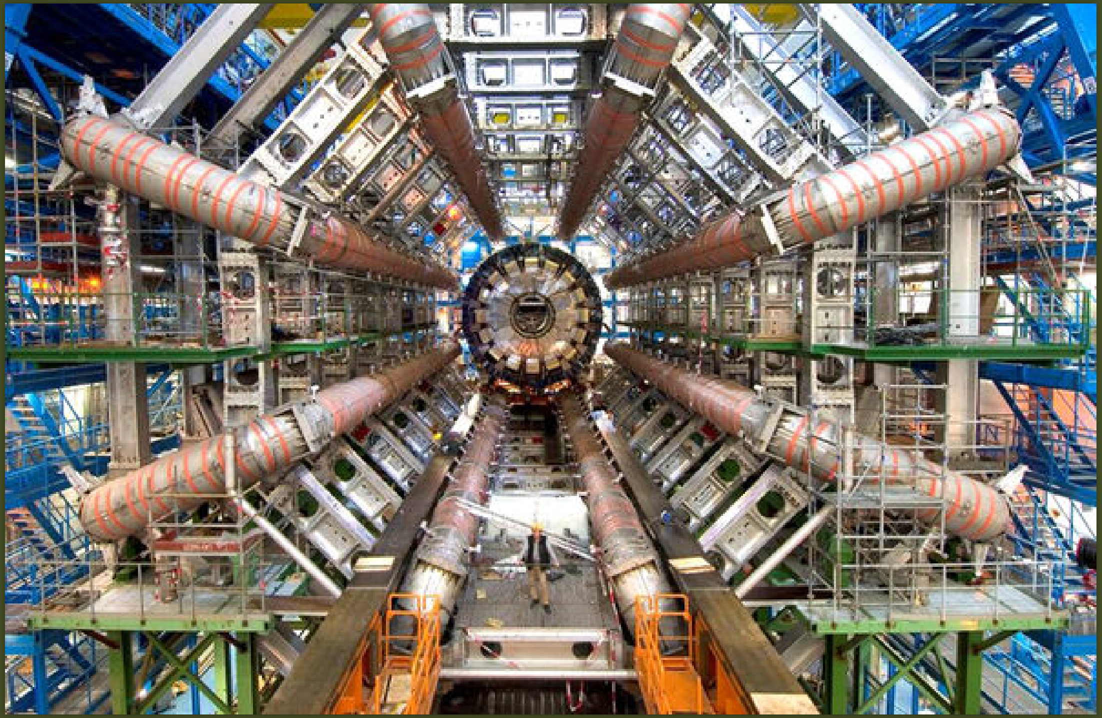


Data: Growth (as of 2013)

- “2.5 Quintillion (or 2.5 million trillion) bytes of data are created every day.”
- “90% of the data that we have right now were created last 2 years alone”

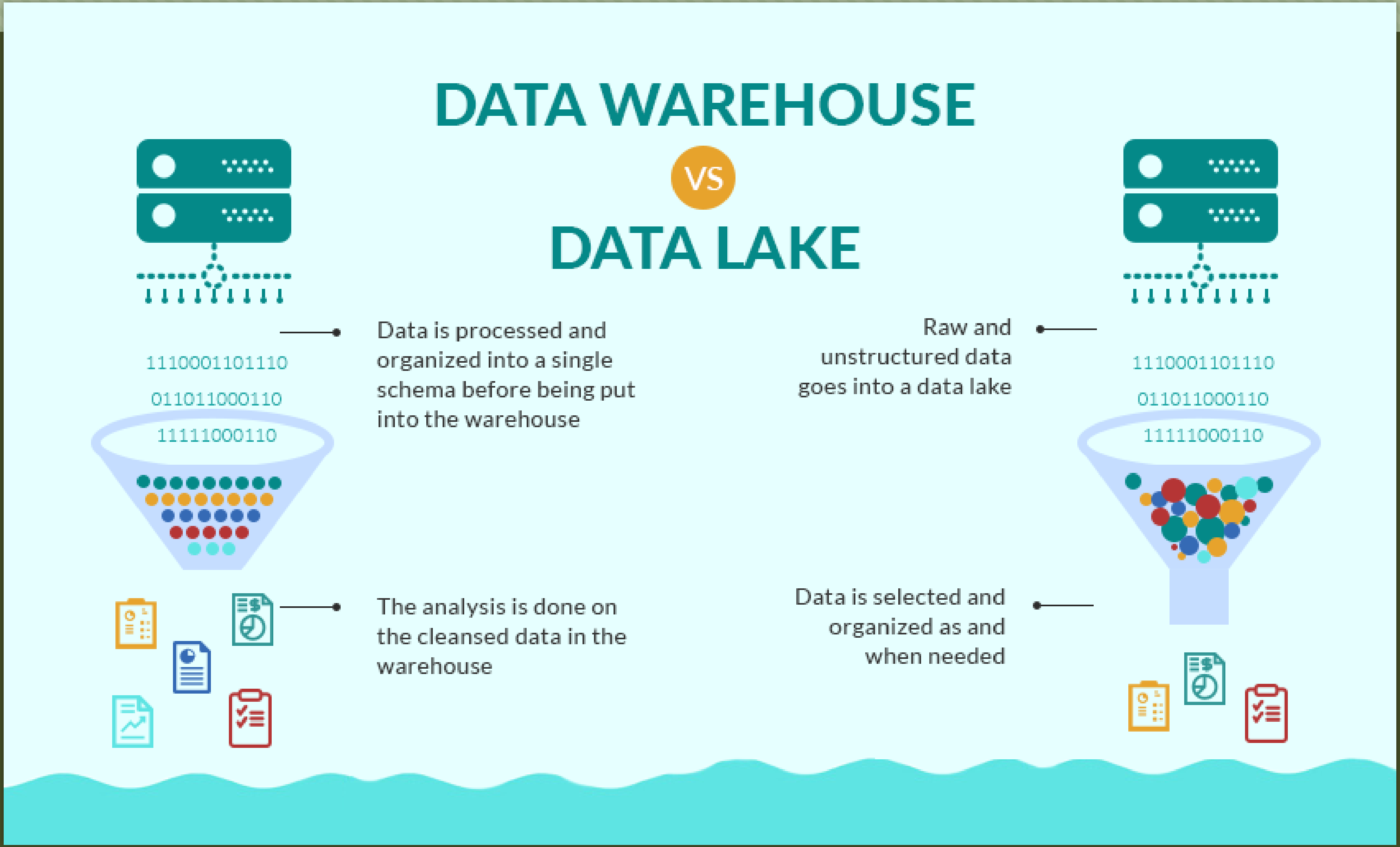
CERN

- LHC experiments produce about 90 petabytes of data per year, and an additional 25 petabytes of data are produced per year for data from other (non-LHC) experiments at CERN



https://home.cern/science/computing/storage

Data Lakes vs Data Warehouse


DATA WAREHOUSE
VS
DATA LAKE
1110001101110
011011000110
11111000110
Data is processed and organized into a single schema before being put into the warehouse
The analysis is done on the cleansed data in the warehouse
Raw and unstructured data goes into a data lake
Data is selected and organized as and when needed

Netflix: House of Cards

- *House of Cards* received positive reviews and several award nominations, including 33 Primetime Emmy Award nominations, including for Outstanding Drama Series, Outstanding Lead Actor for Spacey, and Outstanding Lead Actress for Wright.
- It is the first original online-only streaming television series to receive major Emmy nominations.
- The show also earned eight Golden Globe Award nominations, with Wright winning for Best Actress – Television Series Drama in 2014 and Spacey winning for Best Actor – Television Series Drama in 2015.

NETFLIX

https://en.wikipedia.org/wiki/House_of_Cards_(American_TV_series)

Netflix: House of Cards

- Through big data analytics, Netflix was able to secure its success.
- **Netflix identified that the British version of House of Cards was watched by many subscribers. Those members who watched the British version of House of Cards also seemed to favor movies starring Kevin Spacey**. This was one of the patterns that led to Kevin Spacey being cast in the lead role.
- In fact, big data was instrumental in how most of the characters were cast. It had a role in how the script was finalized and how the overall narrative progressed.

https://sofy.tv/blog/big-data-helped-netflix-series-house-cards-become-blockbuster/

When Data Growth Gets Out of Hand: Big Data

- Velocity – How fast are we generating the data
- Variety – Diversity of data sources
- Volume – How big is the data
- Veracity* – Reliability of the data

Data Mining

- “Extracting knowledge from large amounts of data”
- “Discovery of models* for data”
- *Models = representation(s) of data

Types of Data Mining Tasks

- Descriptive
  - General properties of the data in the DB

- Sample Questions:
  - What is my sales for August?
  - What is my sales for 2020?

Types of Data Mining Tasks

- Predictive
  - Inference on the current data to make predictions

- Sample Questions:
  - What are the chances that this customer will buy item X given that he has bought item Y?
  - What is the language

Types of Data Mining Tasks

- Prescriptive
  - Using data to come up with decisions
  - Data-driven decisions

Data Mining in a Nutshell

“Patterns”

Intro to Docker

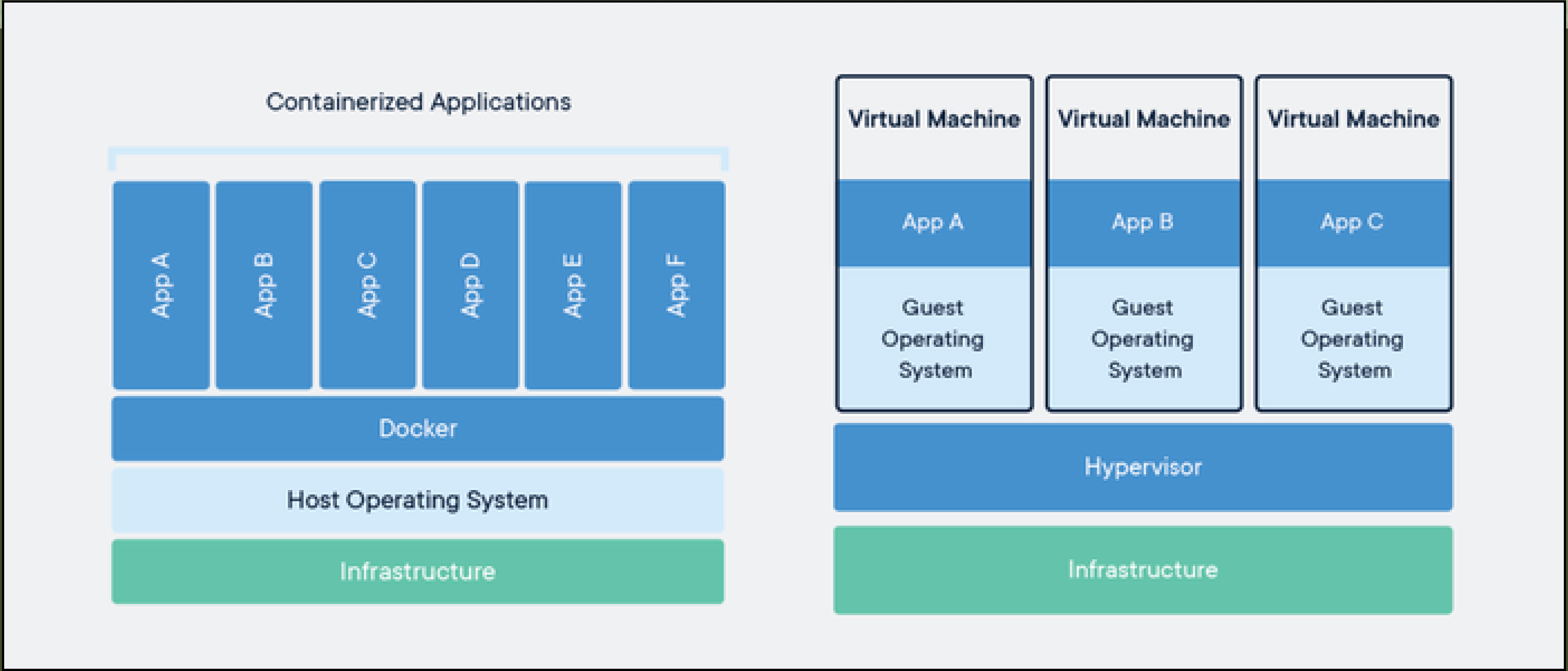
Docker

- Docker is similar in concept to Virtual Machines, except it’s much more lightweight. Instead of running an entire separate operating system (which is a massive overhead), Docker runs containers, which use the same host operating system, and only virtualize at a software level.

Why use Docker?

- Alleviates “it’s broken on my machine!” often caused by configuration / versioning issues
- “Dockerize” your environment and share it with your teammates
  - E.g., you developed an app that uses a web sever, DB, API, etc. (usually in different servers)

Why use Docker?

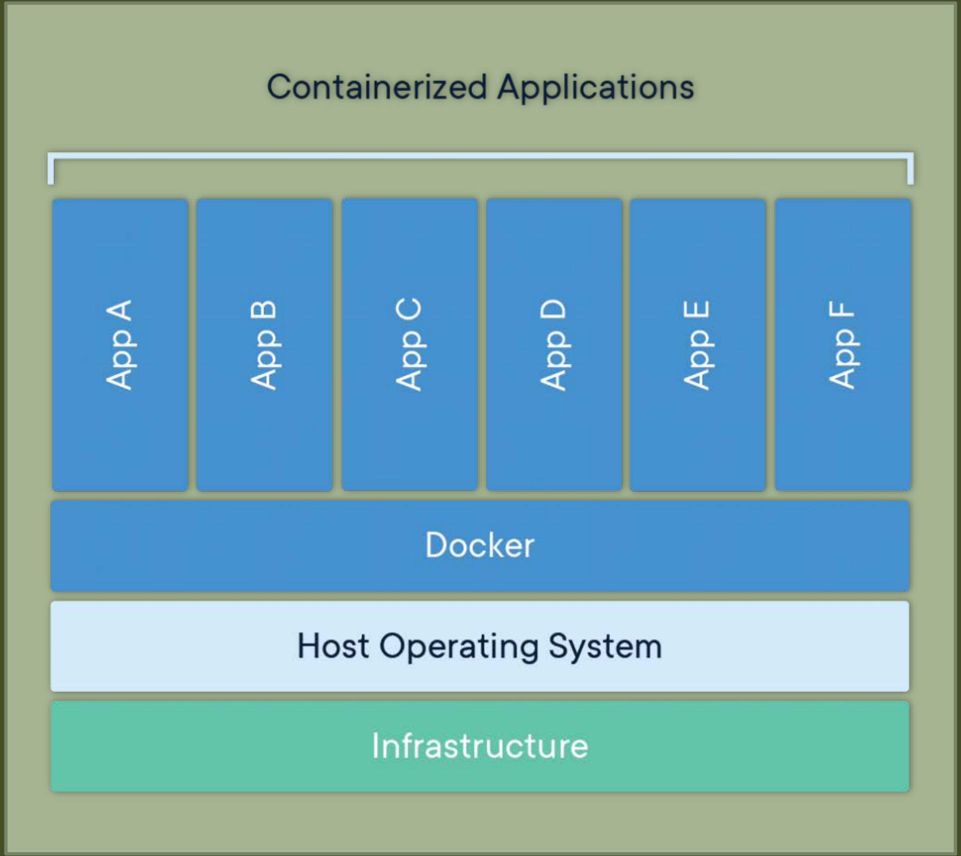


Compatibilities

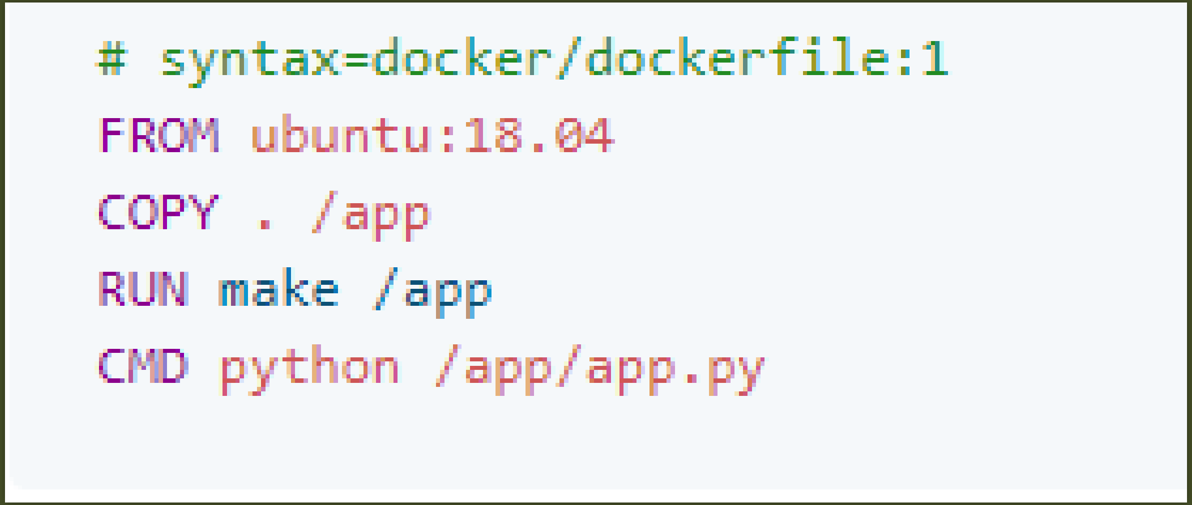
- There are a lot of cloud-based systems that supports docker such as Azure, AWS, DigitalOcean, etc.

How to Use Docker: Quick Start

- Docker Image (template) – file used to execute code in a Docker container. It acts as a set of instructions to build a Docker container.
- Docker Container – a standard unit of software that packages up code and all its dependencies so that applications can be run from one computing environment to another.
- Dockerfile – text document that contains all the commands a user could call on the command line to assemble an image
- Docker-compose – Tool for defining and running multiple containers / docker applications
  - Docker-compose file – text document



Docker File: Quick Start



```
# syntax=docker/dockerfile:1
FROM ubuntu:18.04
COPY . /app
RUN make /app
CMD python /app/app.py
```

my_image.txt

- docker build –f < path to docker file>
  - Docker build –f my_image.txt

How to Use Docker: Quick Start

- Install Docker
- docker pull (Optional)
  - <Image name>[:<Tag>]
- docker run <image>
  - Run custom name (--name)
  - Run interactive mode (-it)
  - Run detached (-d)
  - Run mount volume (-v)
- docker exec
- docker kill
- docker rm

Demo: busybox image, shell (sh)

https://docs.docker.com/engine/reference/commandline/run/#options

Intro to Hadoop

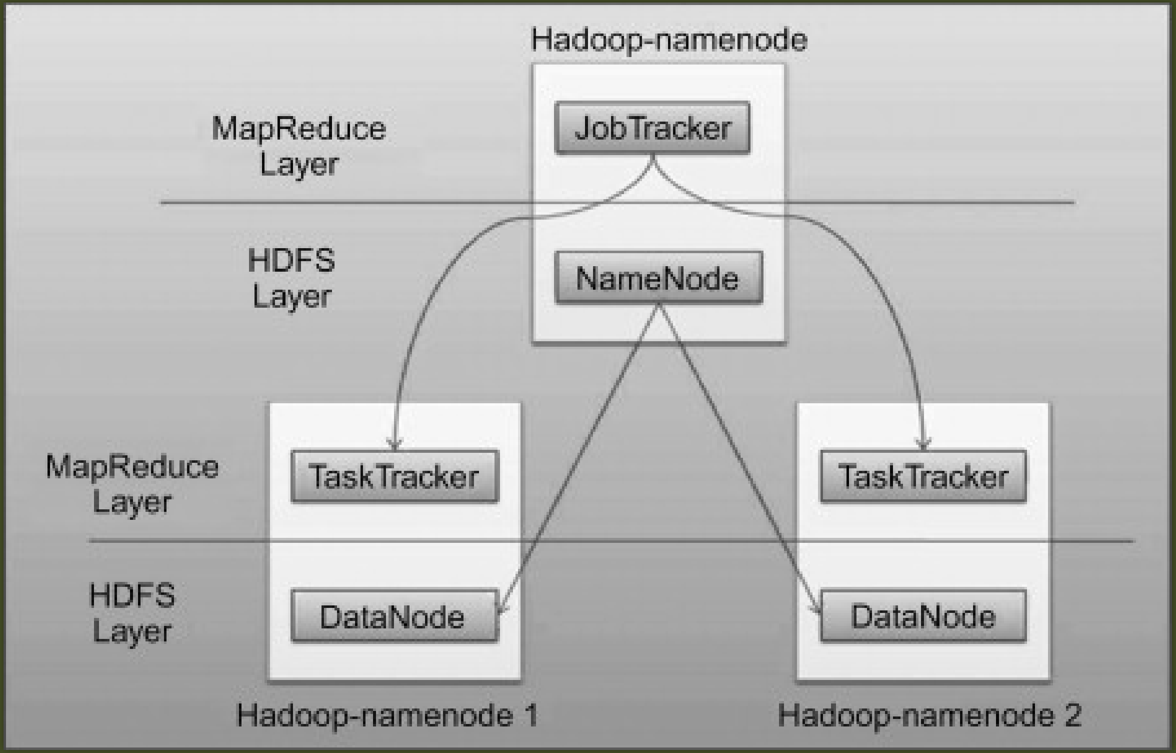
Hadoop

- Hadoop is an open-source framework from Apache that is used to store and analyze data which are very high in volume.
- It is often used for batch processing and used by Facebook, Yahoo, Google, Twitter, etc.
- Commodity Hardware

Capabilities of Hadoop

- Scalable
- Fault-Tolerant
- Store Massive Data Sets
- Mix Disparate Data Sources
- Ingest Bulk Data
- Ingest High Velocity Data
- Apply Structure to Unstructured Data / Semi-structured Data
- Make Data Available for Fast Processing with SQL on Hadoop
- Achieve Data Integration
- Machine Learning and Predictive Analytics
- …

Hadoop Architecture (Physical View)

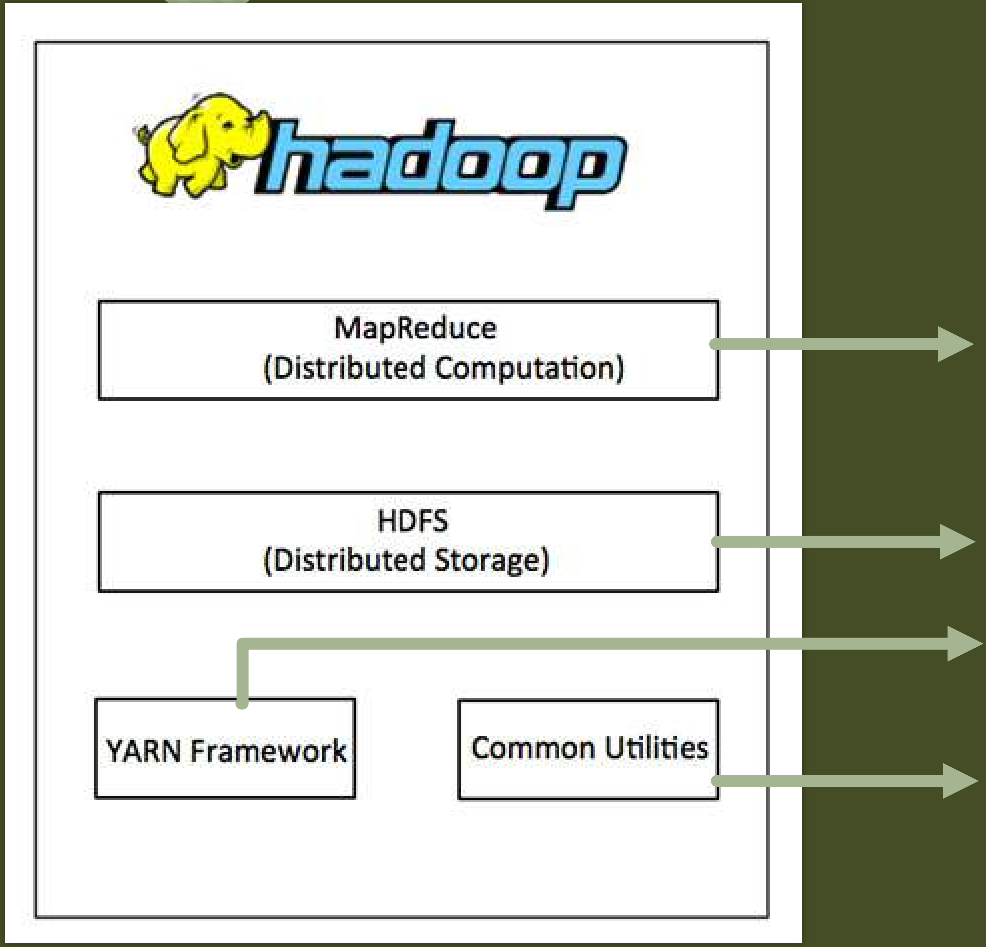


- Name Node – manages the file system, tracks the tasks being done in the cluster, metadata

- Data Node – stores the actual data, where computation happens

- Both Name node and Data Node may run on *commodity hardware*
- Usually GNU / Linux Operating System

Hadoop Architecture (Logical View)



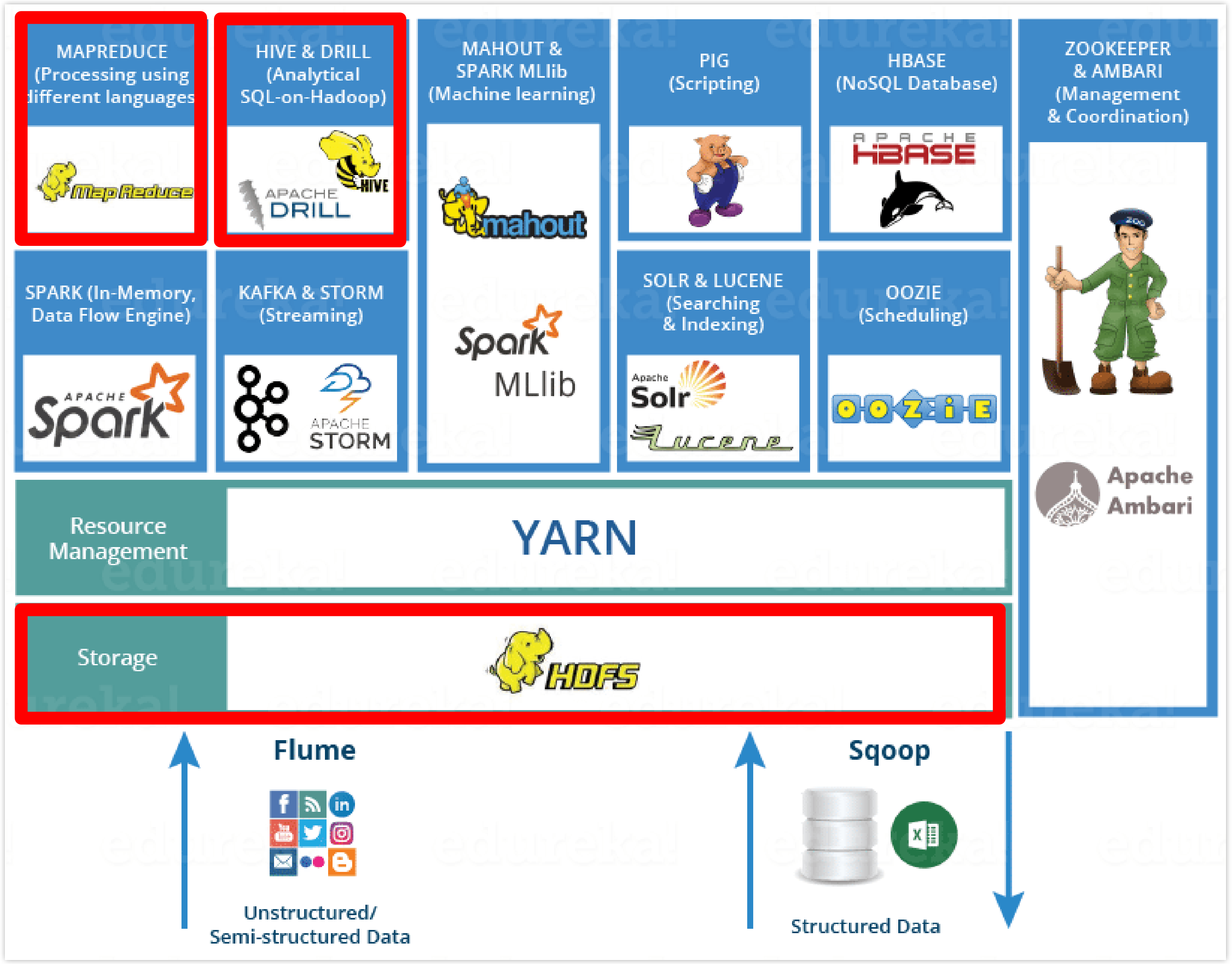
Processing / Computation Layer

Storage Layer

Framework for job scheduling and cluster resource management

Java libraries and utilities required by Hadoop modules

Hadoop Ecosystem


MAPREDUCE (Processing using different languages)
MapReduce
HIVE & DRILL (Analytical SQL-on-Hadoop)
APACHE DRILL
HIVE
MAHOUT & SPARK MLlib (Machine learning)
mahout
Spark MLlib
PIG (Scripting)
HBASE (NoSQL Database)
APACHE HBASE
ZOOKEEPER & AMBARI (Management & Coordination)
Apache Ambari
SPARK (In-Memory, Data Flow Engine)
APACHE Spark
KAFKA & STORM (Streaming)
APACHE STORM
SOLR & LUCENE (Searching & Indexing)
Apache Solr
Lucene
OOZIE (Scheduling)
OOZIE
Resource Management
YARN
Storage
HDFS
Flume
Unstructured/ Semi-structured Data
Sqoop
Structured Data

https://www.edureka.co/blog/hadoop-ecosystem

Hadoop Distributed File System (HDFS) (Hadoop Storage)

- Based on Google File System
- Provides a distributed file system that is designed to run on commodity hardware
- Files are broken in blocks and distributed across the Hadoop *physical* system

Hadoop Distributed File System (HDFS) (Hadoop Storage)

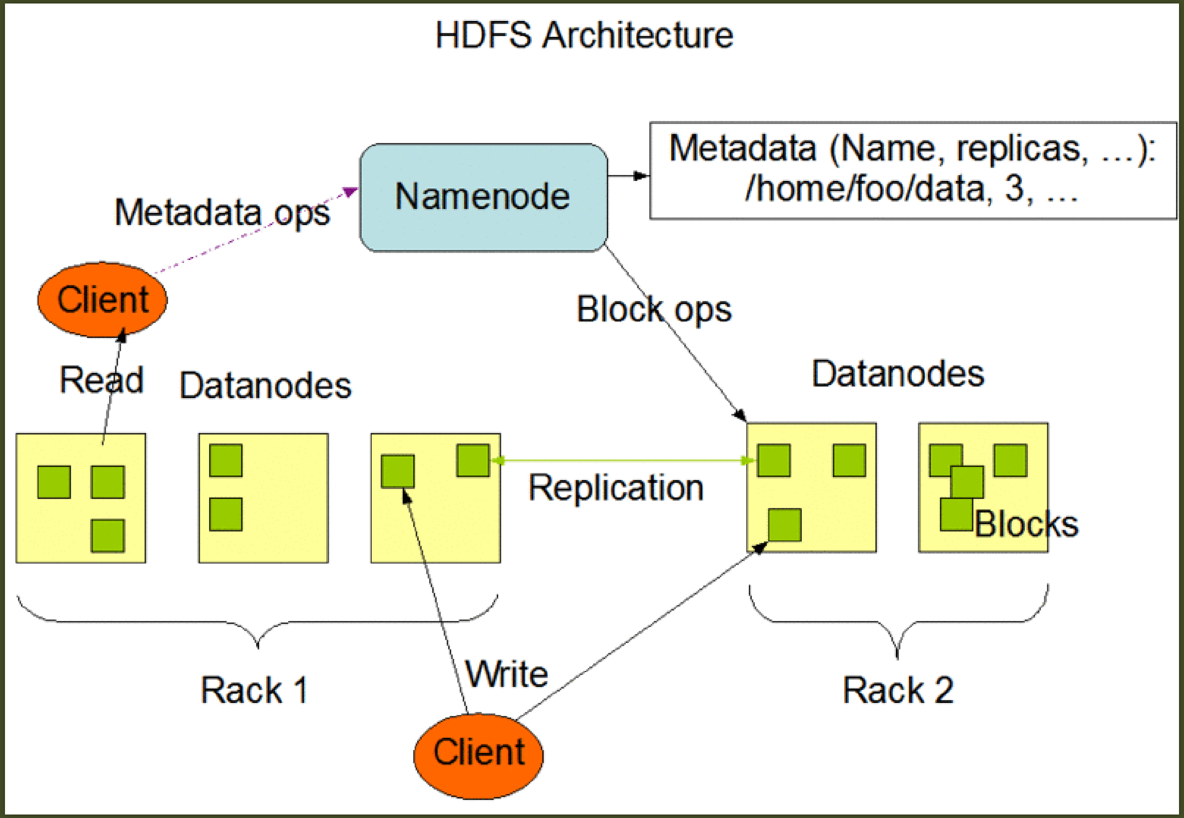
Where to use HDFS

- Very Large Files
- Streaming Data Access
- Commodity Hardware

Where not to use HDFS

- Low latency data access
- Lots of small files
- Multiple writes

Hadoop Distributed File System (HDFS) (Hadoop Storage)



- Default size of block is 64 / 128MB
- Default replication factor of 3
- Replication
- Rack awareness (based on switch)

Server Farm

HDFS – Datanode Heartbeat

- Datanode sends a heartbeat to namenode every 3 seconds
- Datanodes that fail to send a heartbeat to namenode after 10 mins. Is considered a dead node

Interacting with HDFS

- Through the name node
- HDFS Commands

Demo:
docker-compose up
docker cp

Demo:
Hadoop Web UI
Putting files into HDFS

https://www.edureka.co/blog/hdfs-commands-hadoop-shell-command

Inspecting files in HDFS

- hdfs dfsadmin -report
- hdfs fsck input/f1.txt –locations –blocks – files

Demo: Adding 1 more datanode to the cluster (Scaling)

- `docker run –it –d –v docker-hadoop_hadoop_datanodeThree:/hadoop/dfs/data –network docker-hadoop_default --name datanodeThree --env-file ./hadoop.env bde2020/hadoop-datanode:2.0.0-hadoop3.2.1-java8`

- Reinspect namenode GUI
- Reinspect livenodes in hdfs dfsadmin -report
- Reinspect file

https://www.edureka.co/blog/hdfs-commands-hadoop-shell-command

Demo: HDFS Fault Tolerance

- Part 1:
  - Kill datanodeOne
  - Remove datanodeOne volume
  - Access file

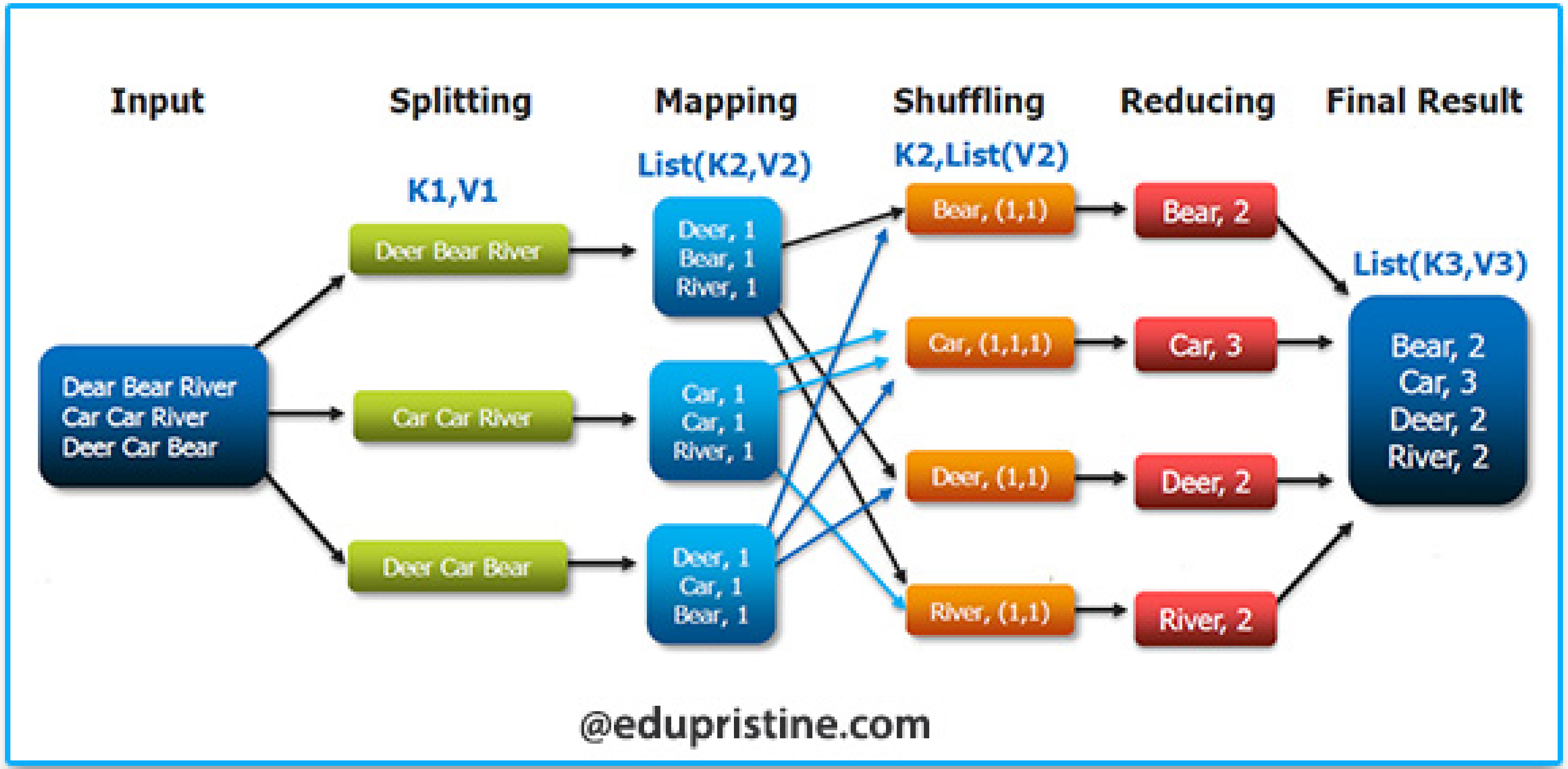
- Part 2:
  - Kill datanodeTwo
  - Remove datanodeOne volume
  - Access file

https://www.edureka.co/blog/hdfs-commands-hadoop-shell-command

Map-Reduce (Distributed Computing)

- MapReduce is a processing technique and a program model for distributed computing based on java. The MapReduce algorithm contains two important tasks, namely Map and Reduce.
  - Map takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key/value pairs).
  - Secondly, reduce task, which takes the output from a map as an input and combines those data tuples into a smaller set of tuples.
- MapReduce is that it is easy to scale data processing over multiple computing nodes
- MapReduce framework operates on <key, value> pairs, that is, the framework views the input to the job as a set of <key, value> pairs and produces a set of <key, value> pairs as the output of the job

Map-Reduce Algorithm in Word Count



Map-Reduce

- Recall that blocks of data are distributed across the cluster.
- This is the basis of which data will be processed by what data node (Map phase)

Running a MR Function (Word Count)

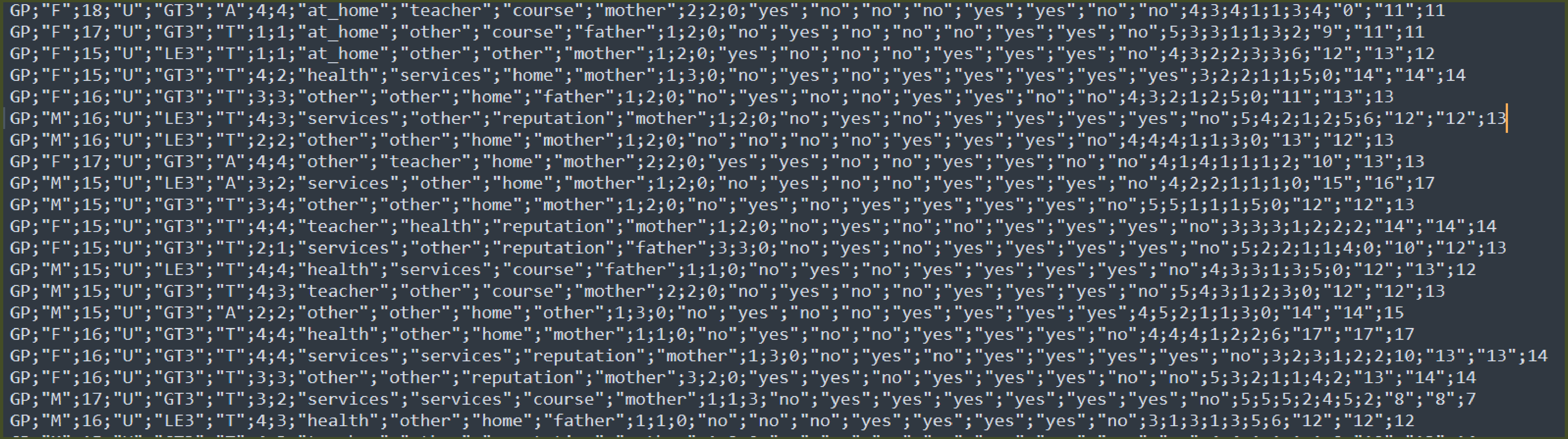
- hadoop jar hadoop-mapreduce-examples-2.7.1-sources.jar org.apache.hadoop.examples.WordCount input output

Sample 1:
MR Function Anatomy (Word Count)

- https://cwiki.apache.org/confluence/display/HADOOP2/WordCount

Sample 2: MR Function (Average)

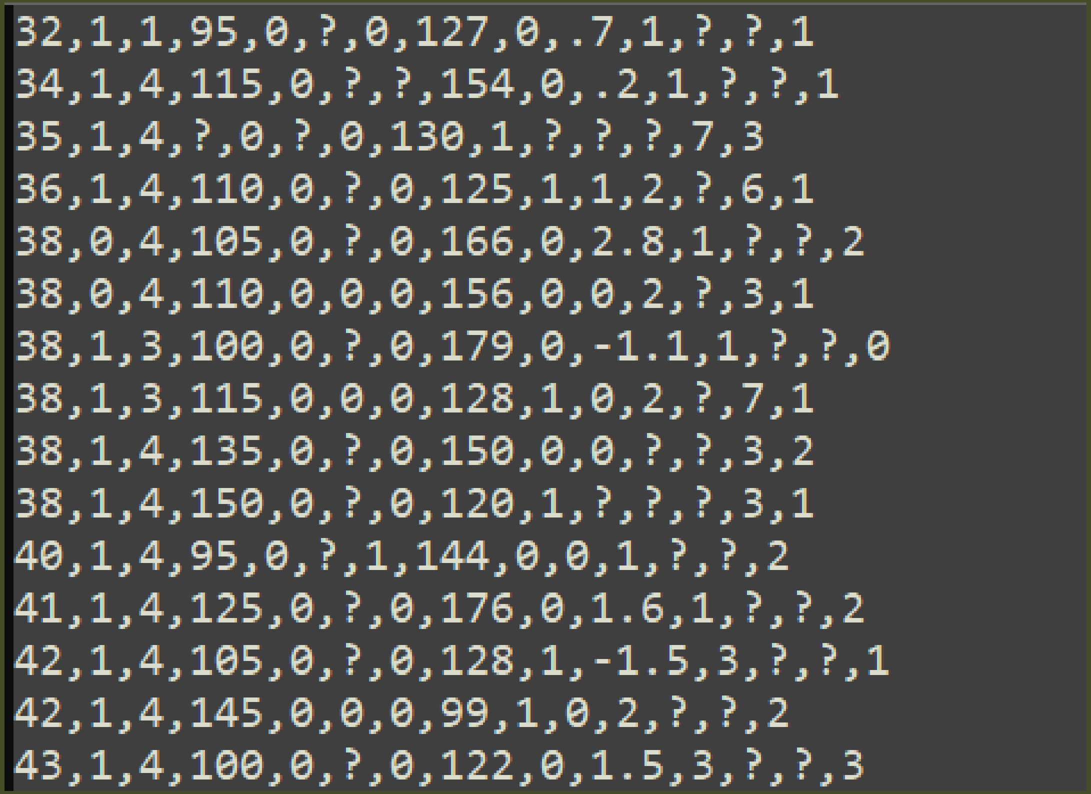
- https://archive.ics.uci.edu/ml/datasets/student+performance#
  - Input -> Preprocessing -> Map -> Shuffle -> Reduce -> Final Output



```
GP;"F";18;"U";"GT3";"A";4;4;"at_home";"teacher";"course";"mother";2;2;0;"yes";"no";"no";"no";"yes";"yes";"no";"no";4;3;4;1;1;3;4;"0";"11";11
GP;"F";17;"U";"GT3";"T";1;1;"at_home";"other";"course";"father";1;2;0;"no";"yes";"no";"no";"no";"yes";"yes";"no";5;3;3;1;1;3;2;"9";"11";11
GP;"F";15;"U";"LE3";"T";1;1;"at_home";"other";"other";"mother";1;2;0;"yes";"no";"no";"no";"yes";"yes";"yes";"no";4;3;2;2;3;3;6;"12";"13";12
GP;"F";15;"U";"GT3";"T";4;2;"health";"services";"home";"mother";1;3;0;"no";"yes";"no";"yes";"yes";"yes";"yes";"yes";3;2;2;1;1;5;0;"14";"14";14
GP;"F";16;"U";"GT3";"T";3;3;"other";"other";"home";"father";1;2;0;"no";"yes";"no";"no";"yes";"yes";"no";"no";4;3;2;1;2;5;0;"11";"13";13
GP;"M";16;"U";"LE3";"T";4;3;"services";"other";"reputation";"mother";1;2;0;"no";"yes";"no";"yes";"yes";"yes";"yes";"no";5;4;2;1;2;5;6;"12";"12";13
GP;"M";16;"U";"LE3";"T";2;2;"other";"other";"home";"mother";1;2;0;"no";"no";"no";"no";"yes";"yes";"yes";"no";4;4;4;1;1;3;0;"13";"12";13
GP;"F";17;"U";"GT3";"A";4;4;"other";"teacher";"home";"mother";2;2;0;"yes";"yes";"no";"no";"yes";"yes";"no";"no";4;1;4;1;1;1;2;"10";"13";13
GP;"M";15;"U";"LE3";"A";3;2;"services";"other";"home";"mother";1;2;0;"no";"yes";"no";"no";"yes";"yes";"yes";"no";4;2;2;1;1;1;0;"15";"16";17
GP;"M";15;"U";"GT3";"T";3;4;"other";"other";"home";"mother";1;2;0;"no";"yes";"no";"yes";"yes";"yes";"yes";"no";5;5;1;1;1;5;0;"12";"12";13
GP;"F";15;"U";"GT3";"T";4;4;"teacher";"health";"reputation";"mother";1;2;0;"no";"yes";"no";"no";"yes";"yes";"yes";"no";3;3;3;1;2;2;2;"14";"14";14
GP;"F";15;"U";"GT3";"T";2;1;"services";"other";"reputation";"father";3;3;0;"no";"yes";"no";"yes";"yes";"yes";"yes";"no";5;2;2;1;1;4;0;"10";"12";13
GP;"M";15;"U";"LE3";"T";4;4;"health";"services";"course";"father";1;1;0;"no";"yes";"no";"yes";"yes";"yes";"yes";"no";4;3;3;1;3;5;0;"12";"13";12
GP;"M";15;"U";"GT3";"T";4;3;"teacher";"other";"course";"mother";2;2;0;"no";"yes";"no";"no";"yes";"yes";"yes";"no";5;4;3;1;2;3;0;"12";"12";13
GP;"M";15;"U";"GT3";"A";2;2;"other";"other";"home";"other";1;3;0;"no";"yes";"no";"no";"yes";"yes";"yes";"yes";4;5;2;1;1;3;0;"14";"14";15
GP;"F";16;"U";"GT3";"T";4;4;"health";"other";"home";"mother";1;1;0;"no";"yes";"no";"no";"yes";"yes";"yes";"no";4;4;4;1;2;2;6;"17";"17";17
GP;"F";16;"U";"GT3";"T";4;4;"services";"services";"reputation";"mother";1;3;0;"no";"yes";"no";"yes";"yes";"yes";"yes";"no";3;2;3;1;2;2;10;"13";"13";14
GP;"F";16;"U";"GT3";"T";3;3;"other";"other";"reputation";"mother";3;2;0;"yes";"yes";"no";"yes";"yes";"yes";"no";"no";5;3;2;1;1;4;2;"13";"14";14
GP;"M";17;"U";"GT3";"T";3;2;"services";"services";"course";"mother";1;1;3;"no";"yes";"yes";"yes";"yes";"yes";"yes";"no";5;5;5;2;4;5;2;"8";"8";7
GP;"M";16;"U";"LE3";"T";4;3;"health";"other";"home";"father";1;1;0;"no";"no";"no";"yes";"yes";"yes";"yes";"no";3;1;3;1;3;5;6;"12";"12";12
```

Sample 3: MR Function (Min & Max Value)

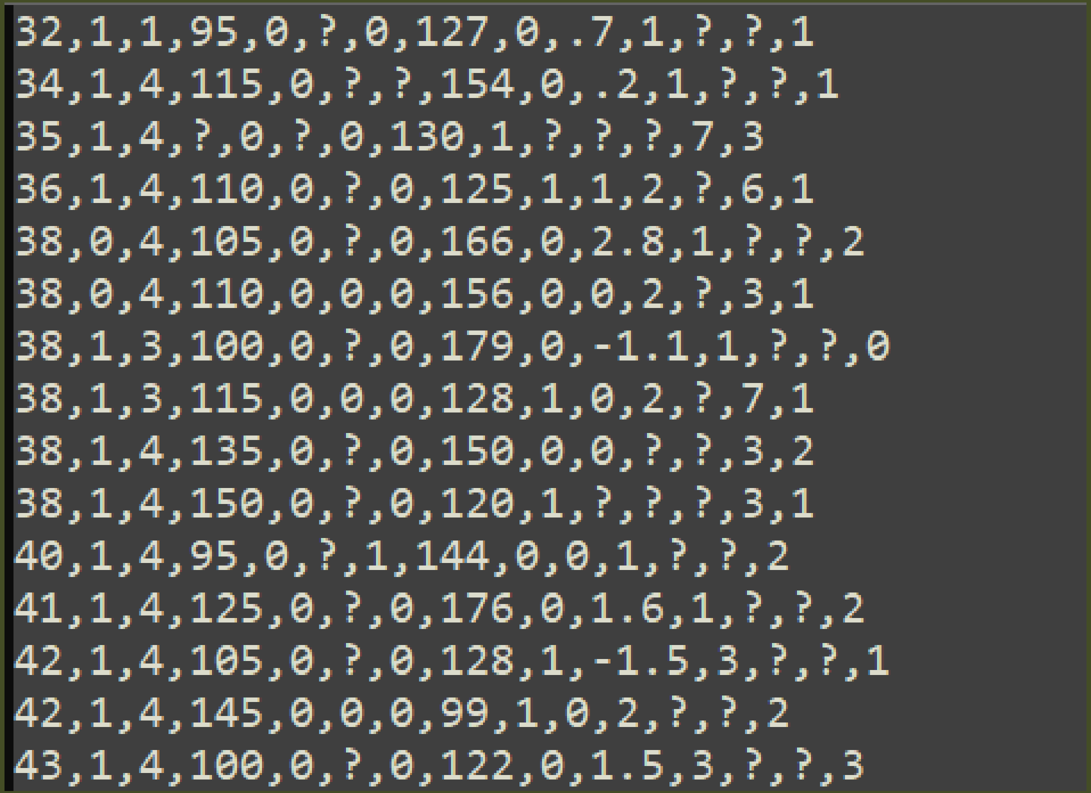
- https://archive.ics.uci.edu/ml/machine-learning-databases/heart-disease/processed.switzerland.data



```
32,1,1,95,0,?,0,127,0,.7,1,?,?,1
34,1,4,115,0,?,?,154,0,.2,1,?,?,1
35,1,4,?,0,?,0,130,1,?,?,?,7,3
36,1,4,110,0,?,0,125,1,1,2,?,6,1
38,0,4,105,0,?,0,166,0,2.8,1,?,?,2
38,0,4,110,0,0,0,156,0,0,2,?,3,1
38,1,3,100,0,?,0,179,0,-1.1,1,?,?,0
38,1,3,115,0,0,0,128,1,0,2,?,7,1
38,1,4,135,0,?,0,150,0,0,?,?,3,2
38,1,4,150,0,?,0,120,1,?,?,?,3,1
40,1,4,95,0,?,1,144,0,0,1,?,?,2
41,1,4,125,0,?,0,176,0,1.6,1,?,?,2
42,1,4,105,0,?,0,128,1,-1.5,3,?,?,1
42,1,4,145,0,0,0,99,1,0,2,?,?,2
43,1,4,100,0,?,0,122,0,1.5,3,?,?,3
```

Hands-on Exercise: MR Function (Ave. Cholesterol by Sex)

- https://archive.ics.uci.edu/ml/machine-learning-databases/heart-disease/processed.switzerland.data



```
32,1,1,95,0,?,0,127,0,.7,1,?,?,1
34,1,4,115,0,?,?,154,0,.2,1,?,?,1
35,1,4,?,0,?,0,130,1,?,?,?,7,3
36,1,4,110,0,?,0,125,1,1,2,?,6,1
38,0,4,105,0,?,0,166,0,2.8,1,?,?,2
38,0,4,110,0,0,0,156,0,0,2,?,3,1
38,1,3,100,0,?,0,179,0,-1.1,1,?,?,0
38,1,3,115,0,0,0,128,1,0,2,?,7,1
38,1,4,135,0,?,0,150,0,0,?,?,3,2
38,1,4,150,0,?,0,120,1,?,?,?,3,1
40,1,4,95,0,?,1,144,0,0,1,?,?,2
41,1,4,125,0,?,0,176,0,1.6,1,?,?,2
42,1,4,105,0,?,0,128,1,-1.5,3,?,?,1
42,1,4,145,0,0,0,99,1,0,2,?,?,2
43,1,4,100,0,?,0,122,0,1.5,3,?,?,3
```

Hadoop Streaming

- Hadoop Streaming is a utility that comes with the Hadoop distribution.
- Hadoop streaming can be performed using languages like Python, Java, PHP, Scala, Perl, UNIX, and many more.
- The utility allows us to create and run Map/Reduce jobs with any executable or script as the mapper and/or the reducer.

Hadoop Streaming

```
HADOOP_HOME/bin/hadoop  jar $HADOOP_HOME/hadoop-streaming.jar
-input myInputDirs
-output myOutputDir
-mapper /bin/cat
-reducer /bin/wc
```

Hadoop Streaming (MAP)

mapper.py

```
#!/usr/bin/python
import  sys
#Word Count Example
# input comes from standard input STDIN
for  line in  sys.stdin:
    line =  line.strip() #remove leading and trailing whitespaces
    words =  line.split() #split the line into words and returns as a list
    for  word in  words:
      #write the results to standard output STDOUT
      print'%s    %s'  %  (word,1) #Emit the word
```

Hadoop Streaming (Reduce)

reducer.py

```
#!/usr/bin/python
import sys
from operator import itemgetter
# using a dictionary to map words to their counts
current_word = None
current_count = 0
word = None
# input comes from STDIN
for line in sys.stdin:
      line = line.strip()
      word,count = line.split('   ',1)
      try:
           count = int(count)
      except ValueError:
           continue
      if current_word == word:
           current_count += count
      else:
           if current_word:
               print '%s   %s' % (current_word, current_count)
           current_count = count
           current_word = word
if current_word == word:
      print '%s   %s' % (current_word,current_count)
```

Intro to Hive

Hive

- Hive is a data warehouse infrastructure tool to process structured data in Hadoop

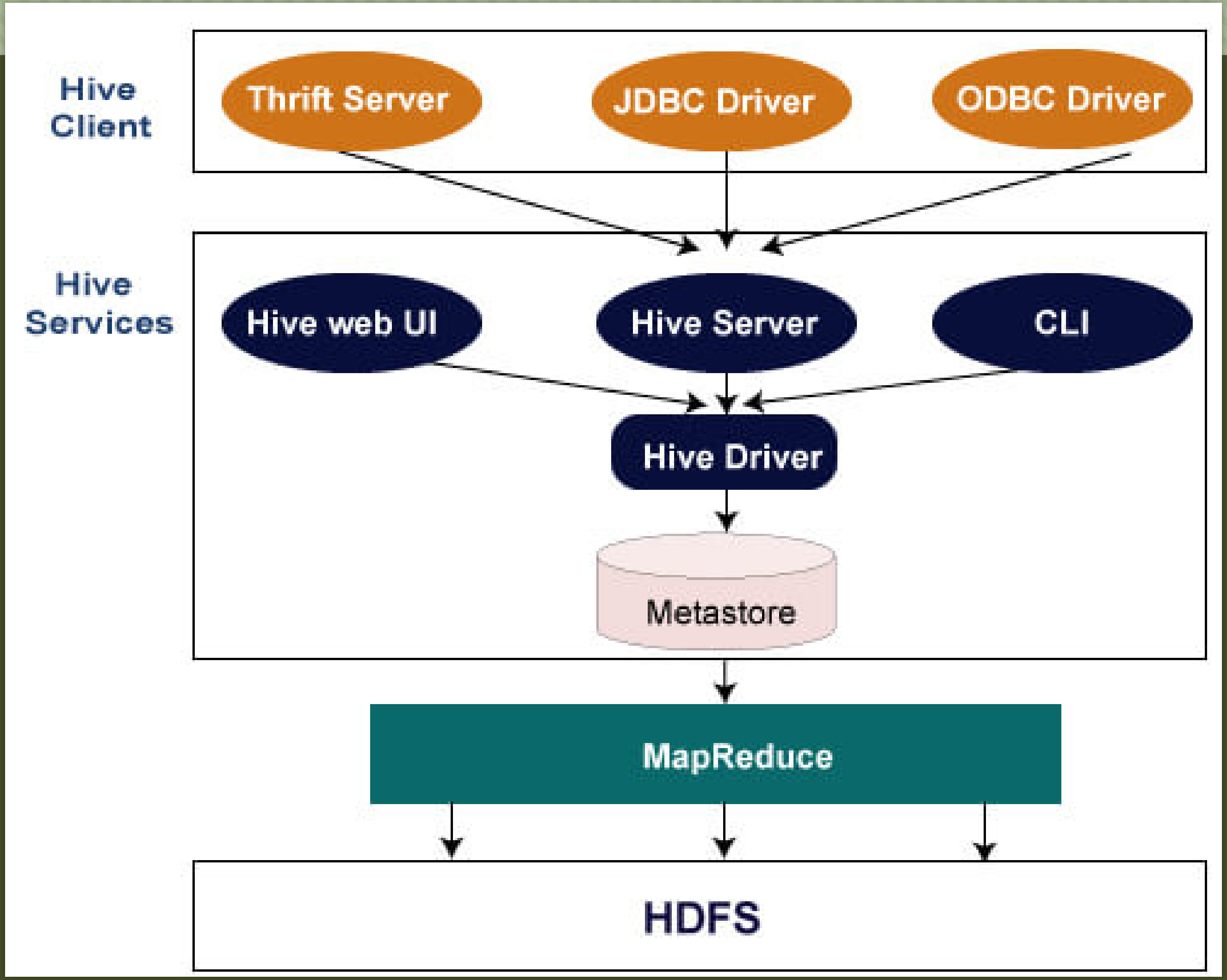
Hive is not:

- A relational database
- A design for Online Transaction Processing (OLTP)
- A language for real-time queries and row-level updates

Features of Hive

- It stores schema in a database and processed data into HDFS.
- It is designed for OLAP.
- It provides SQL type language for querying called HiveQL or HQL.
- It is familiar, fast, scalable, and extensible.

Hive Architecture



Hive Create Table

```
CREATE [TEMPORARY] [EXTERNAL] TABLE
[IF NOT EXISTS] [db_name.]
table_name

[(col_name data_type [COMMENT
col_comment], ...)]

[COMMENT table_comment]

[ROW FORMAT row_format]

[STORED AS file_format]
```

```
CREATE TABLE IF NOT EXISTS employee
( eid int, name String, salary
String, destination String)

COMMENT ‘Employee details’

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ‘\t’

LINES TERMINATED BY ‘\n’

STORED AS TEXTFILE;
```

https://www.tutorialspoint.com/hive/hive_partitioning.htm

Hive Partitions

- Hive organizes tables into partitions. It is a way of dividing a table into related parts based on the values of partitioned columns such as date, city, and department. Using partition, it is easy to query a portion of the data.
- Tables or partitions are sub-divided into **buckets,** to provide extra structure to the data that may be used for more efficient querying. Bucketing works based on the value of hash function of some column of a table.

https://www.tutorialspoint.com/hive/hive_partitioning.htm

Sample 1: Word Count in Hive

```
CREATE TABLE FILES (line STRING);

LOAD DATA INPATH 'docs' OVERWRITE INTO TABLE FILES;

CREATE TABLE word_counts AS
SELECT word, count(1) AS count
FROM (SELECT explode(split(line, ' ‘))
      AS word FROM FILES) w
GROUP BY word ORDER BY word;
```

Staging Table

Data Ingestion

Data Processing

https://riptutorial.com/hive/example/21838/word-count-example-in-hive

Sample 1: Word Count in Hive

- Examine created table in namenode GUI

Sample 2: Average in Hive

- https://archive.ics.uci.edu/ml/datasets/student+performance#

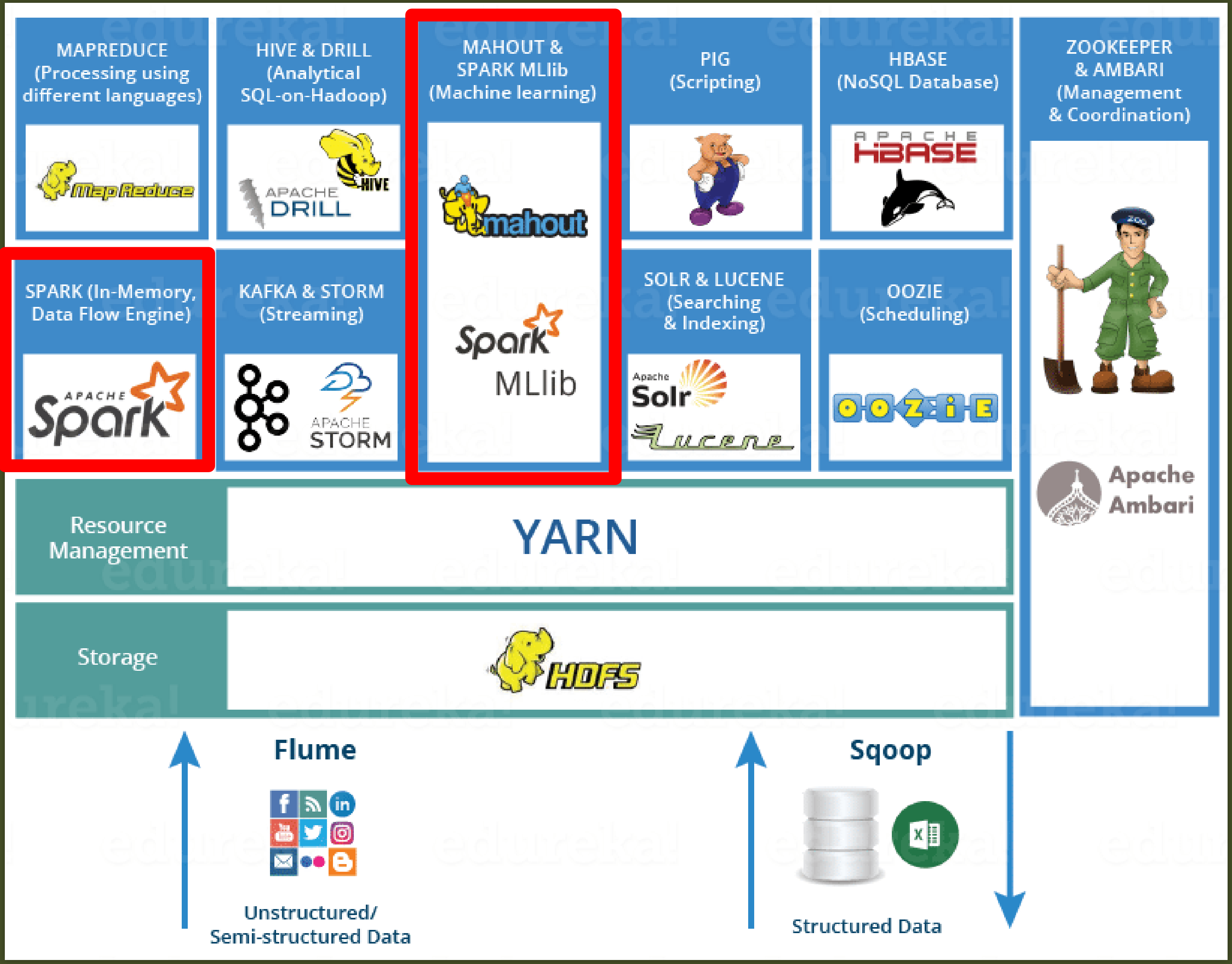
Sample 3: Mix & Max Age Value in Hive

- https://archive.ics.uci.edu/ml/machine-learning-databases/heart-disease/processed.switzerland.data

Hands-on: Average, Highest and Lowest Cholesterol Values by Sex in Hive

- https://archive.ics.uci.edu/ml/machine-learning-databases/heart-disease/processed.switzerland.data

Hadoop Ecosystem


MAPREDUCE (Processing using different languages)
Map Reduce
HIVE & DRILL (Analytical SQL-on-Hadoop)
HIVE
APACHE DRILL
MAHOUT & SPARK MLlib (Machine learning)
mahout
Spark MLlib
PIG (Scripting)
HBASE (NoSQL Database)
APACHE HBASE
ZOOKEEPER & AMBARI (Management & Coordination)
Apache Ambari
SPARK (In-Memory, Data Flow Engine)
APACHE Spark
KAFKA & STORM (Streaming)
APACHE STORM
SOLR & LUCENE (Searching & Indexing)
Apache Solr
Lucene
OOZIE (Scheduling)
OOZIE
Resource Management
YARN
Storage
HDFS
Flume
Unstructured/ Semi-structured Data
Sqoop
Structured Data

https://www.edureka.co/blog/hadoop-ecosystem

End of Presentation