DATALEARNS24+7

YAVA

Data Management Platform

Yava Data Management Platform Essentials

Chapter 01

Big Data Overview

The Rise of Big Data

- Technology Growth
- Internet Adoption
- People Behaviour
- Digitize Everything
- Competition

3S Problem

What is Big Data ?

Buzz phrase, **no single definition**

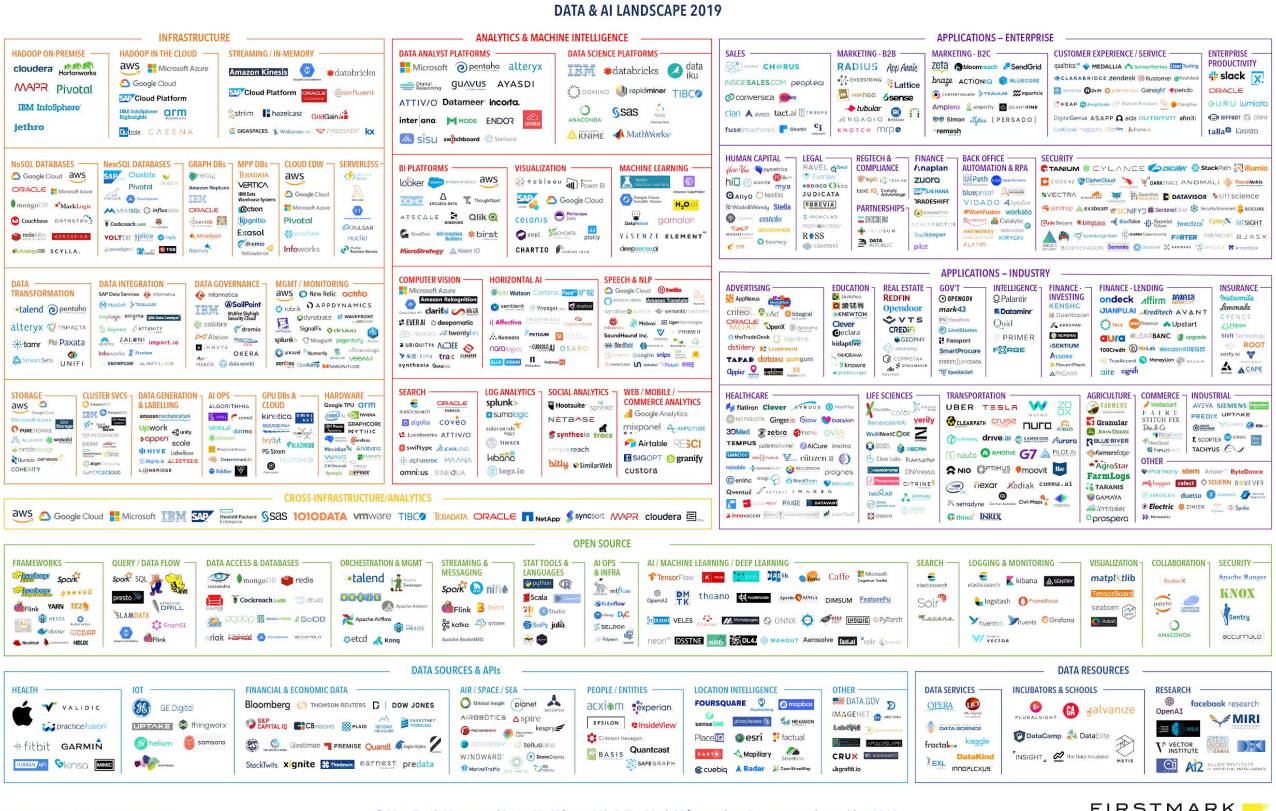
Wikipedia:

> Big data is a term for data sets that are ***so large or complex*** that traditional data processing application software is inadequate to deal with them. Challenges include ***capture, storage, analysis, data curation, search, sharing, transfer, visualization, querying, updating and information privacy***.

Now, refer to Big Data Analytics

Matt Turck

VC at FirstMark Capital. Previously, Managing Director at Bloomberg Ventures and co-founder of TripleHop Technologies. Occasional angel investor. Startup mentor (Techstars, DreamIt, ERA, FGVN, NYC Venture Fellows). Organizer of two large monthly tech community events, Data Driven NYC and Hardwired NYC



source:
https://mattturck.com/data2019

Big Data and AI Landscape

Matt Turck
VC at FirstMark Capital
mattturck.com/bigdata2019

INFRASTRUCTURE

Infrastructure

Hadoop on Prem, Hadoop on Cloud, Streaming/In Memory, NoSQL Databases, NewSQL Databases, Graph DBs, MPP DBs, Cloud EDW, Data Transformation, Data Integration, Data Governance, Mgmt/Monitoring, Storage, Cluster Svcs, App Dev, Crowd Sourcing, Hardware, GPU DBs

ANALYTICS

Analytics

Data Analytics Platforms, Data Science Platforms, BI Platforms, Visualization, Machine Learning, Computer Vision, Horizontal AI, Speech & NLP, Search, Log Analytics, Social Analytics, Web/Mobile/Commerce Analytics

APPLICATIONS – ENTERPRISE

Application Enterprise

Sales - Marketing B2B - Marketing B2C - Customer Service - Human Capital - Legal - Finance - Enterprise Productivity - Back Office Automation - Security

APPLICATIONS – INDUSTRY

Application Industry

Advertising - Education - Government
Finance - Real Estate - Insurance - Healthcare
Life Science - Transportation - Agriculture
Commerce - etc

CROSS-INFRASTRUCTURE/ANALYTICS

Cross Infrastructure/Analytics

OPEN SOURCE

Open Source

Frameworks, Query/Data Flow, Data Access, Coordination, Streaming, Stat Tools, AI/ML/DL, Search, Logging & Monitoring, Visualization, Collaboration, Security

DATA SOURCES & APIS

Data Source & APIS

Health, IoT, Financial & Economic Data, Air/Space/Sea, People, Location Intelligence, Other

DATA RESOURCES

Data Source

Open Source Technology

- Most of big data component is open source
- We can download the code, use and modify freely
- Require adequate human resources
- Lots of choices

Disruptive Technology

Since it first appearance to the current, big data has changed the existing and established business model.

- Open source – zero license
- Proven by big internet company
- Active community
- Fast adoption

Proven by Big Internet Company

- Invented by internet company
- Used in production environment
- Shared to open source community
- Specific function

Active Community

- Many developers are involved in technology development
- Supported and sponsored by big company

DATA is the new OIL

We don't have better algorithms,
we just have more data

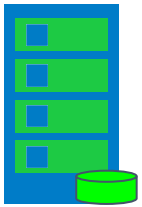
- Peter Norvig - Director of Research at Google

Chapter 02

YAVA Data Management Platform

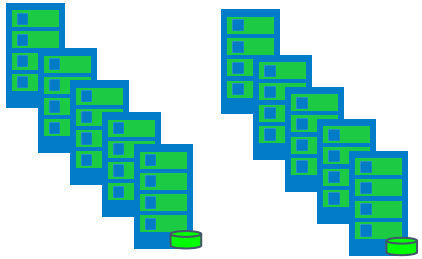
Parallel Processing Basic Concept

→ **Read 1 TB Data**



1 Machine
- 4 I/O channels
- each channel : 100 MB/s

Theoretically : **45 minutes**
- 250 GB per channel



10 Machine
- 4 I/O channels
- each channel : 100 MB/s

Theoretically : **4.5 minutes**
- 25 GB per channel

Parallel Processing is Complex

- Job balancing
- Hardware failure and failover
- Most analysis tasks need to be able to combine the data

What is Hadoop ?

- Open Source Platform for data management
- Combination of distributed storage and distributed processing
- Computer cluster built from commodity hardware
- Framework written in java programming
- Offering scalability and high performance
- The name Hadoop is not an acronym
- Doug Cutting named it after his son's toy elephant

History of Hadoop

- Mike Cafarella and Doug Cutting started the Nutch project in 2002
- In 2003, Google published Google File System paper, that described the architecture of Google's distributed file system
- By adopting GFS, Nutch Distributed File System (NDFS) began to be implemented on the Nutch project in 2004
- In 2004, Google published the paper that introduced MapReduce to the world
- Early in 2005, the Nutch developers had a working MapReduce implementation in Nutch
- In February 2006 they moved out of Nutch to form an independent subproject of Lucene called Hadoop
- April 2006 – Hadoop 0.1.0 was released

Who use Hadoop ?

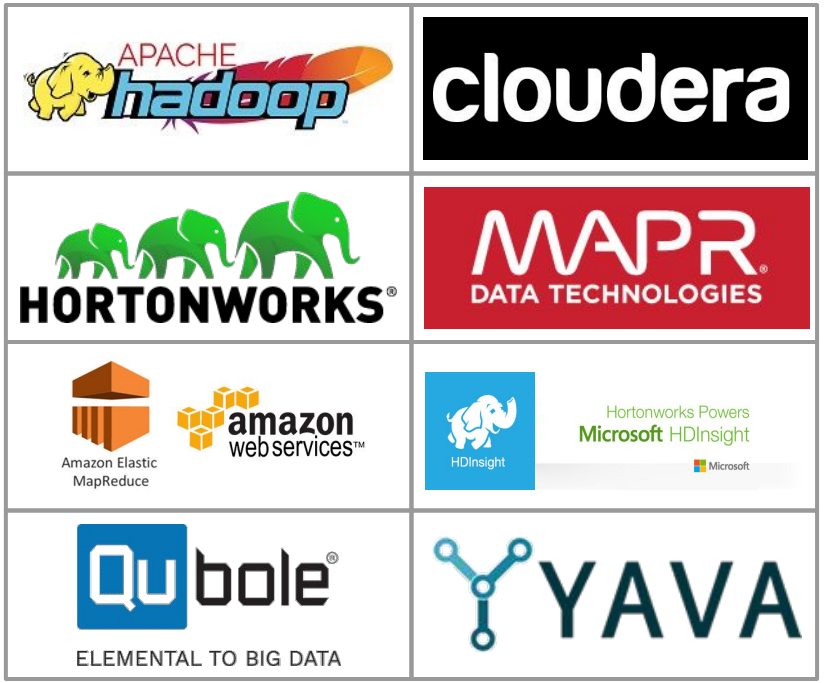
In 2008, Yahoo! Inc. the world's largest Hadoop production application, runs on a Linux cluster with more than 10,000 cores. Now more than 100,000 CPUs in > 40,000 computers running Hadoop.

In 2010, Facebook claimed that they had the largest Hadoop cluster in the world with 21 PB of storage, In June 2012, they announced the data had grown to 100 PB, now the data grow 0.5 PB every day

Spotify, 1650 node cluster : 43,000 virtualized cores, ~70TB RAM, ~65 PB storage

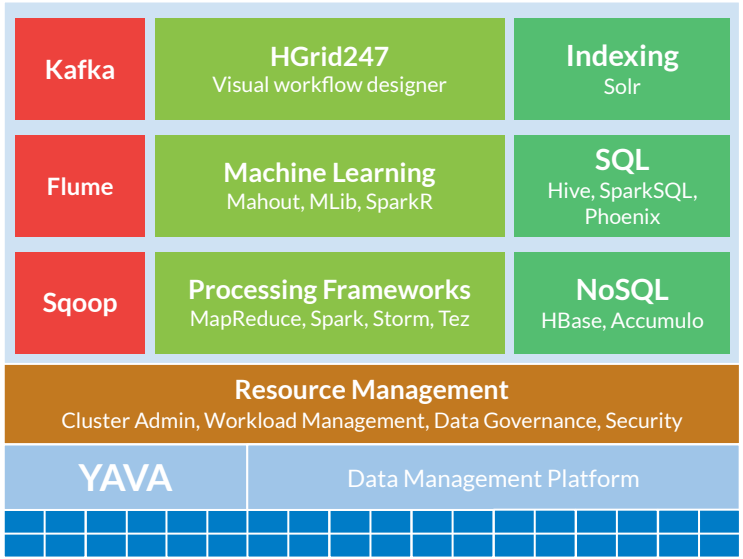
Source: wiki.apache.org/hadoop/PoweredBy

Hadoop Distribution



Magic Quadrant for Data Management Solutions for Analytics

YAVA Data Management Platform



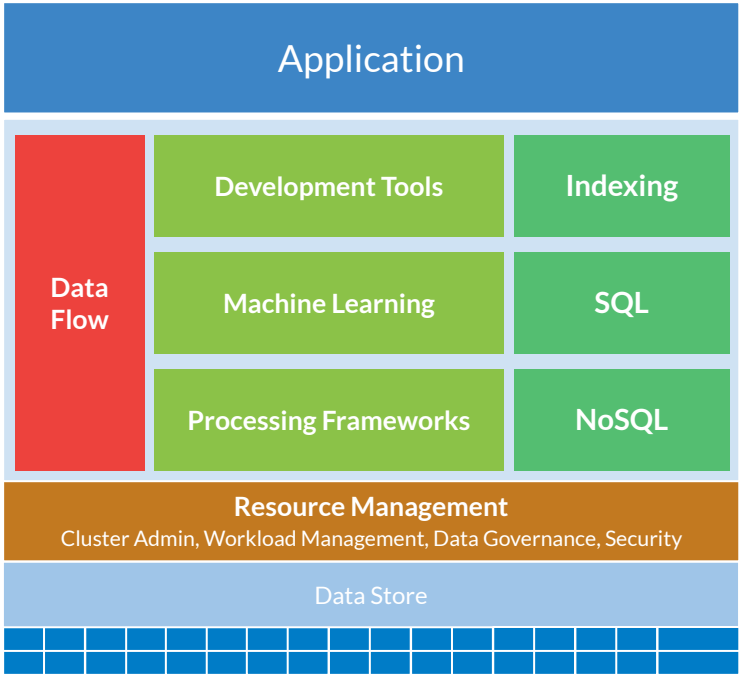
All in one data management platform
Programming/Scripting :

- Java, Python, Scala, R
- SQL
- HGrid247 - Visual Designer

For further info : **yava.labs247.id**

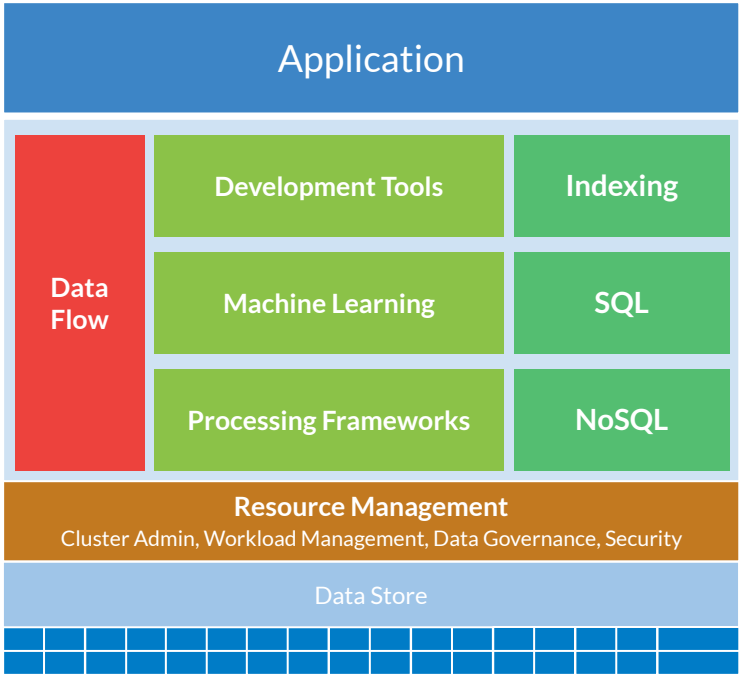
Big data and artificial intelligence platform based on open source component. It is designed to make organization easier to implement big data.

Yava Component



Data Store	HDFS, GlusterFS
Cluster Management	Ambari
Data Governance	Atlas, Falcon
Security	Knox, Ranger
Workload Management	Yarn, Zookeeper, OOzie, Slider
Data Flow	Flume, Sqoop, Kafka, NiFi
Processing Frameworks	MapReduce, Spark, Storm, Tez
Machine Learning	Mahout, MlLib, SparkR, H2O
Development Tools	HGrid247, Zeppelin, Jupyter

Yava Component



NoSQL	HBase, Accumulo
SQL	Hive, Phoenix, Hawq
Indexing	Solr, Elasticsearch
Visualization	Kibana, Superset
Application	Chanthel, Medan, Metron

Yava Community Edition

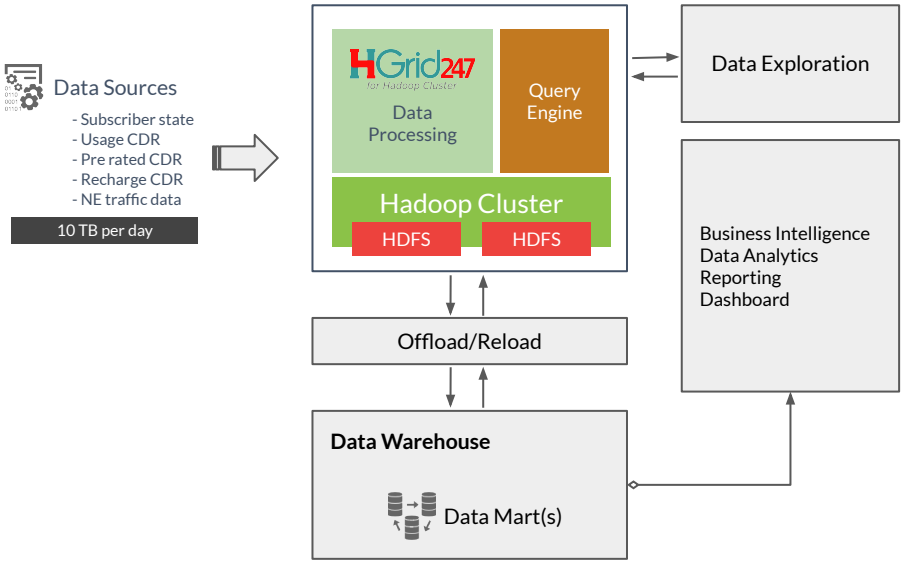
- Based on YAVA247 Data Management Platform
- For:
  - Educational
  - POC/Trial
  - Learning
  - Research

- Community Support
  yava.labs247.id/download_box

Usage

- Archival and Storage
  - Retain years of data
  - Retain intermediate format
- Transformation
  - Map inputs and outputs where needed
  - Turn unstructured data into structured at runtime
- Analysis
  - Explore data in-place
  - Execute arbitrary code

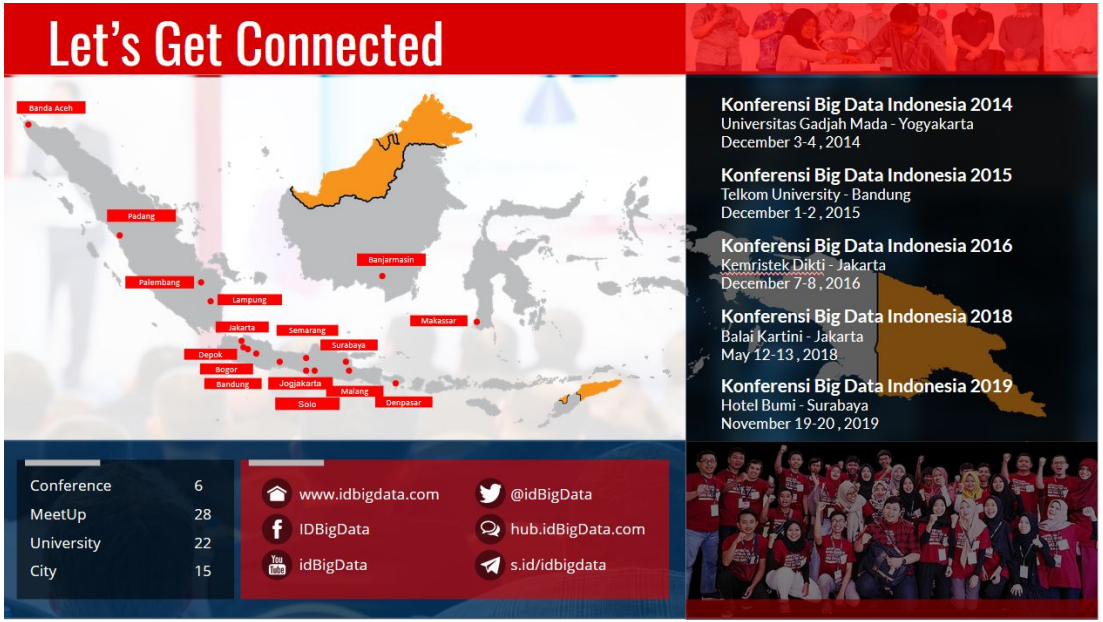
Use Case : Data Warehouse



- Data Warehouse as a single point of truth
- Problem :
  - Cannot achieve SLA
  - Hi Cost

DATA LAKE

idBigData - Komunitas Big Data Indonesia

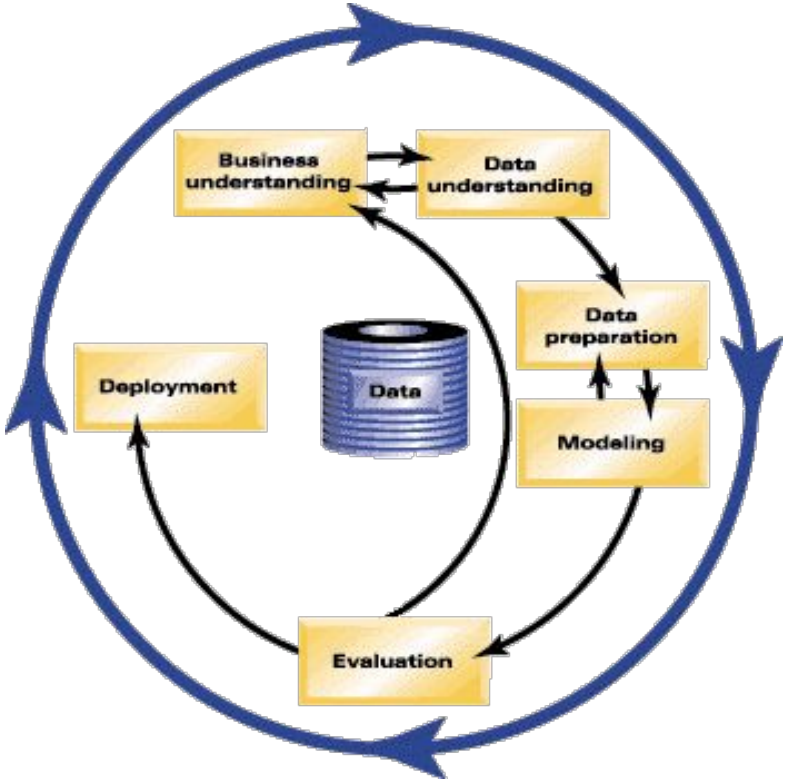


www.idbigdata.com
@idBigData
IDBigData
hub.idBigData.com
idBigData
s.id/idbigdata

Chapter 03

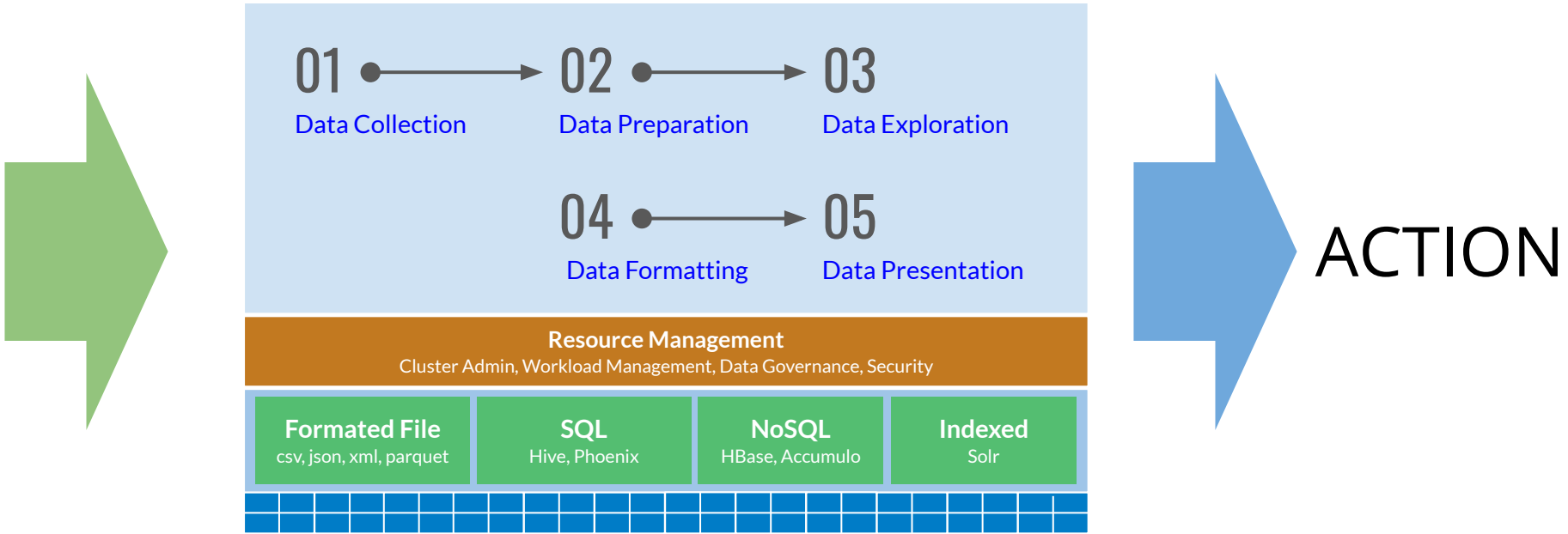
Big Data Processing Workflow

CRISP DM Framework



- Cross-Industry Standard Process for Data Mining, is an industry-proven way to guide your data mining efforts
- Business Understanding : where the business problem is defined and characterised
- Data Understanding and Data Preparation **consume 85% of the total project time**

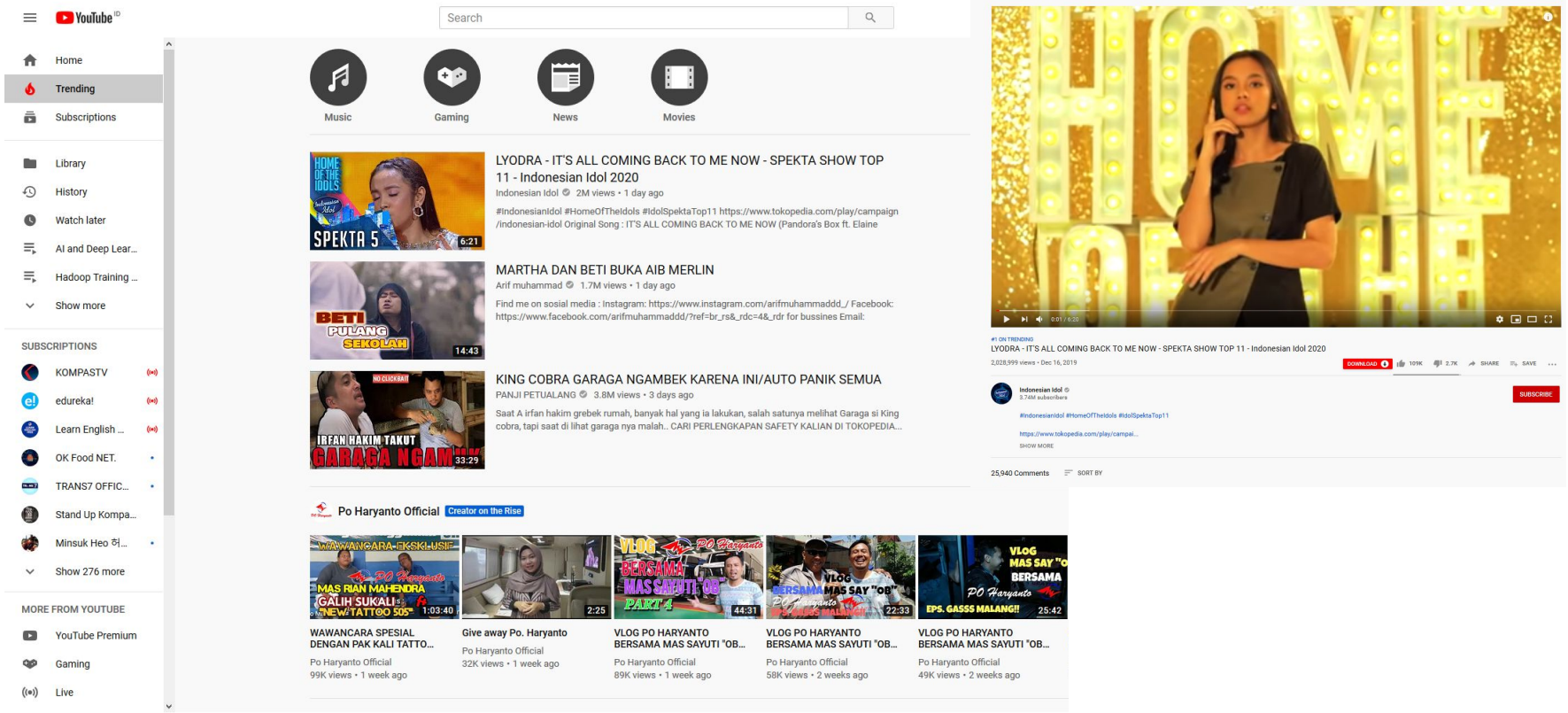
Data Journey



Chapter 04

Business Understanding

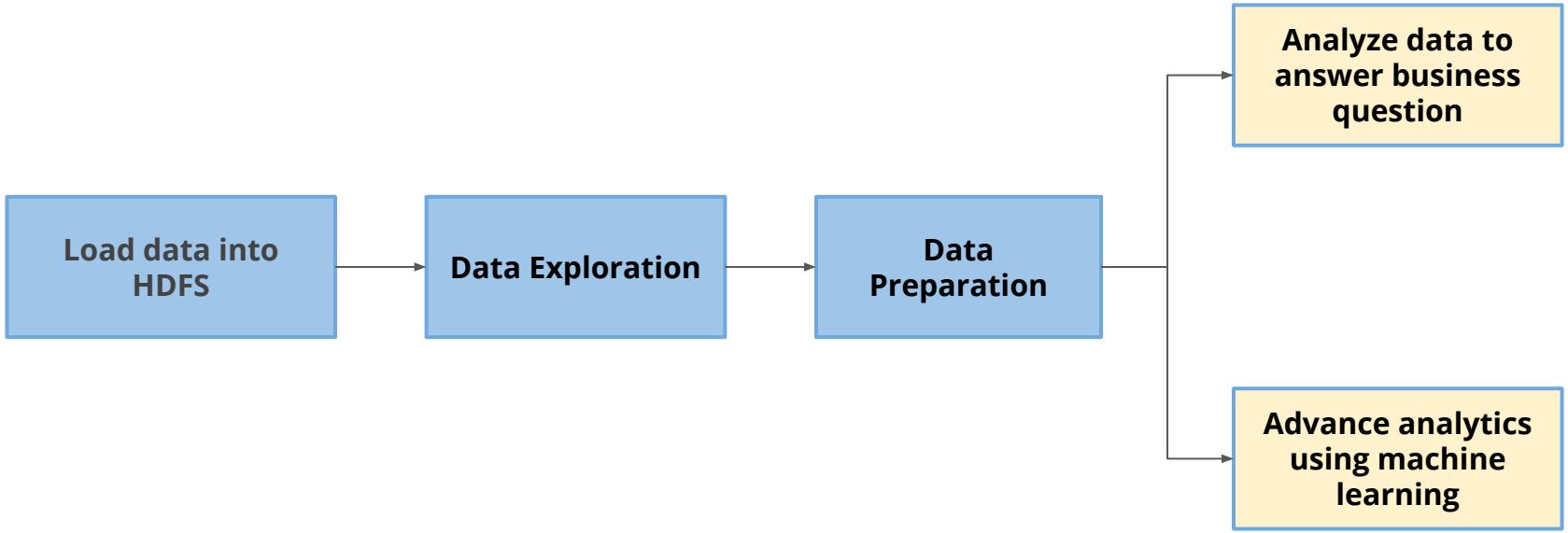
YouTube Video Trending



Business Question

- How long usually a video can trend ?
- How many likes, dislikes, views and comments get ?
- Correlation of trending video in between countries
- Videos from which category has longer trend?
- Users like videos from which CATEGORY the most?
- What is the ratio of Likes-Dislikes and Views-Comments in different categories?

Processing Pipeline



Chapter 05

HDFS - Distributed Storage

Hadoop Component

Main Hadoop Component :

1. **Hadoop Distributed File System**
   a distributed file system designed to run on commodity hardware

2. **MapReduce**
   a software framework for easily writing applications which process vast amounts of data (multi-terabyte data-sets) in-parallel on large clusters (thousands of nodes) of commodity hardware in a reliable, fault-tolerant manner.

3. **Yarn**
   the resource management layer for the Apache Hadoop ecosystem that allows multiple data processing engines such as interactive SQL, real-time streaming, data science and batch processing to handle data stored in a single platform

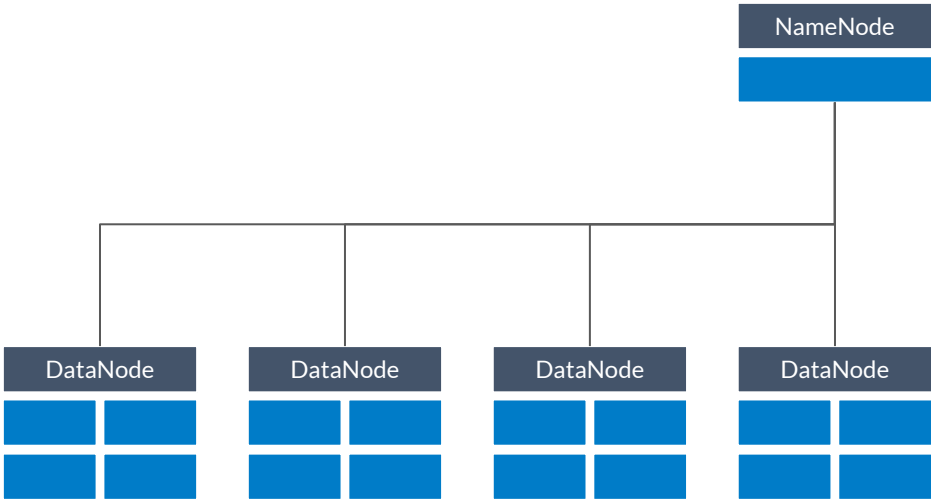
Master Slave Architecture

NameNode

1. Master service
2. Maintain and manage DataNodes
3. Records metadata i.e file size, location of blocks stored, permission, hierarchy, etc
4. Receives heartbeat and block report from DataNodes

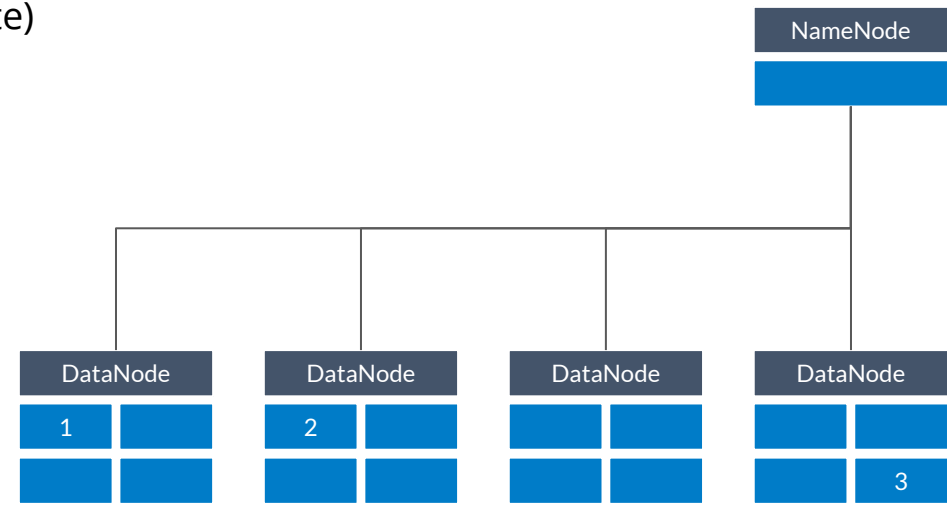
DataNode

1. Slave service
2. Stores physical data
3. Serves read and write requests from clients



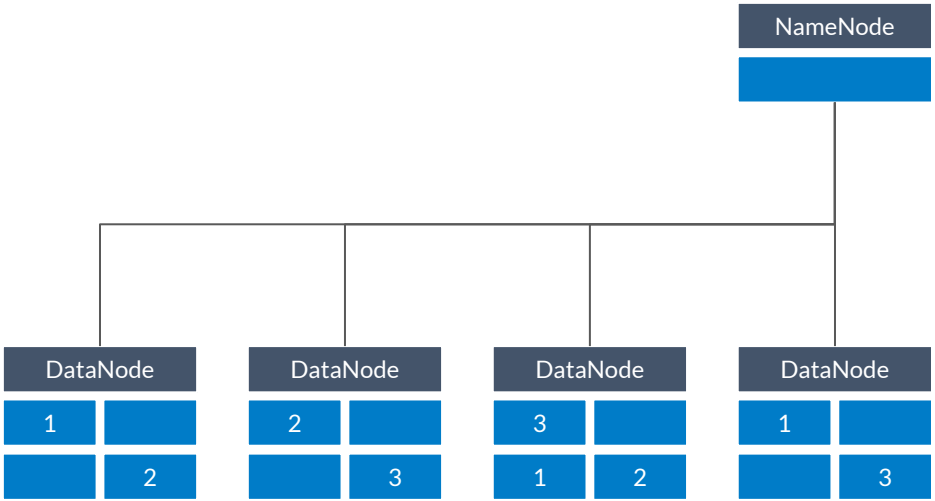
HDFS Block

- HDFS splits huge files into small chunks known as data blocks
- We (client and admin) do not have any control over the data block like block location
- Default size of each block is 128 MB (64 MB in Hadoop 1.x)
- Configuration : HDFS -> Configs -> Advanced -> Advanced hdfs-site
  - dfs.blocksize = 134217728 (in byte)
- Sample.txt → file size : 320 MB
  - Block 1 : 128 MB
  - Block 2 : 128 MB
  - Block 3 : 64 MB



HDFS Replication

- Each block will be replicated
- Default replication is 3
- Configuration : HDFS -> Configs -> Advanced -> General
  - dfs.replication = 3
- Sample.txt → file size : 320 MB
  - Block 1 : 128 MB
  - Block 2 : 128 MB
  - Block 3 : 64 MB



Hands-On

Basic Linux Command

Get Information

1. View OS information

```
# lsb_release -a
```

```
# uname -a
```

2. Verify connected user

```
# whoami
```

3. Get current location

```
# pwd
```

4. List file and directory in current location

```
# ls -la
```

Download Dataset

1. Create new directory

```
# mkdir -p mydata
```

```
# ls -lah
```

2. Download file from dropbox

```
# wget -P mydata https://www.dropbox.com/s/04rm49jqsz0qwrh/kerawang-bekasi.txt
```

3. Verify the file

```
# ls -lah mydata
```

4. View content and count number of line

```
# cat mydata/kerawang-bekasi.txt
```

```
# wc -l mydata/kerawang-bekasi.txt
```

Hands-On

Upload Data Into HDFS

Upload Data into HDFS

1. List file and directory in HDFS user home directory

```
# hdfs dfs -ls
```

2. Create dataset directory

```
# hdfs dfs -mkdir dataset
```

```
# hdfs dfs -ls
```

3. Upload data into HDFS

```
# hdfs dfs -put mydata/kerawang-bekasi.txt dataset
```

4. Verify uploaded data

```
# hdfs dfs -ls dataset
```

Upload Data into HDFS

5. View dataset

```
# hdfs dfs -cat dataset/kerawang-bekasi.txt
```

6. Count number of line

```
# hdfs dfs -cat dataset/kerawang-bekasi.txt | wc -l
```

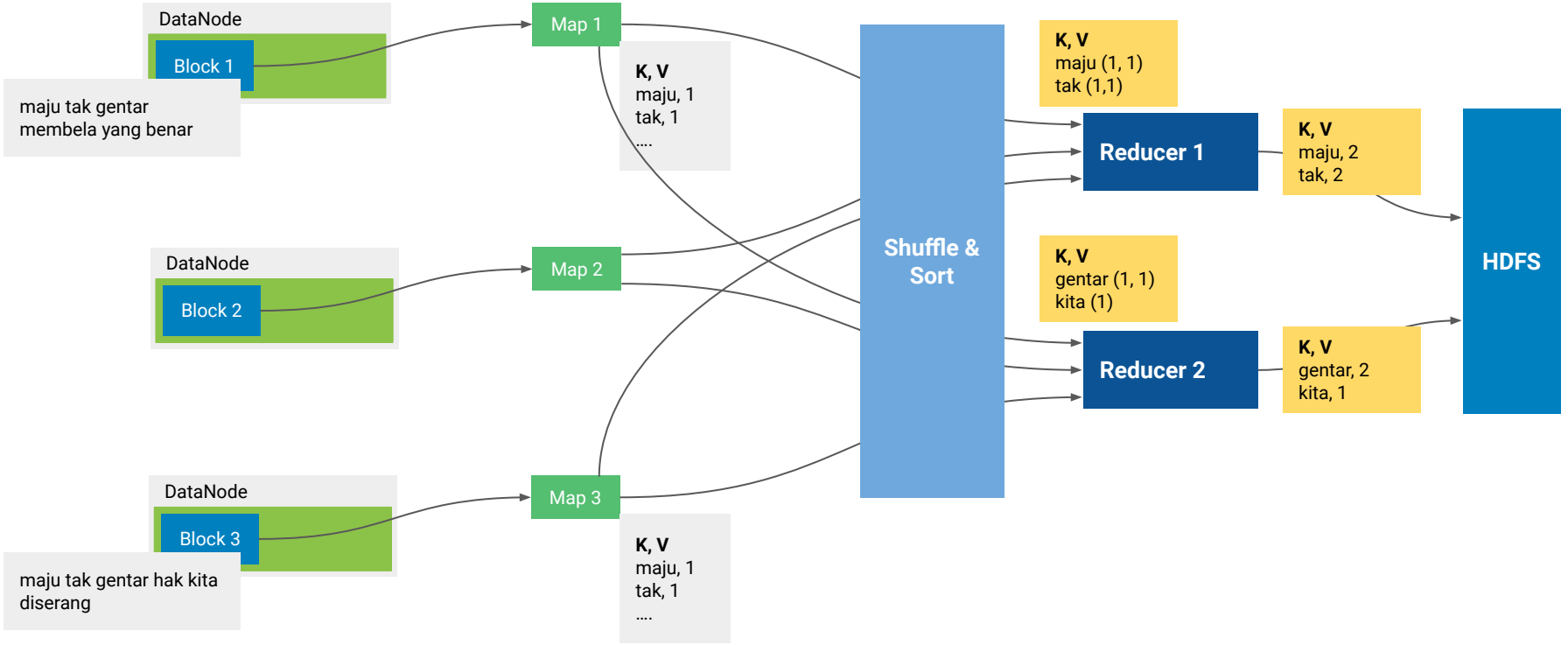
Chapter 06

YARN and MapReduce

Yet Another Resource Negotiator (YARN)

- Introduced in Hadoop 2.x.
- Allows different data processing engines like graph processing, interactive processing, stream processing as well as batch processing to run and process data stored in HDFS
- Manage Resources:
  - scheduling
  - resources assignments (CPU and memory)
- Component:
  - Resource Manager
  - Node Manager (one per worker node)
  - Application Master (one per application)

MapReduce


DataNode
Block 1
maju tak gentar
membela yang benar
DataNode
Block 2
DataNode
Block 3
maju tak gentar hak kita
diserang
Map 1
Map 2
Map 3
K, V
maju, 1
tak, 1
....
K, V
maju, 1
tak, 1
....
Shuffle &
Sort
K, V
maju (1, 1)
tak (1,1)
K, V
gentar (1, 1)
kita (1)
Reducer 1
Reducer 2
K, V
maju, 2
tak, 2
K, V
gentar, 2
kita, 1
HDFS

Hands-On

WordCount

Run Wordcount Application

1. Verify dataset

```
# hdfs dfs -ls dataset
```

2. Execute wordcount program

```
# yarn jar /usr/yava/current/hadoop-mapreduce-client/hadoop-mapreduce-examples.jar wordcount
dataset/kerawang-bekasi.txt wordcount-result
```

3. View the result

```
# hdfs dfs -ls wordcount-result
```

```
# hdfs dfs -cat wordcount-result/part-r-00000
```

4. Delete file and directory on HDFS

```
# hdfs dfs -rm -r wordcount-result
```

Chapter 07

Hive - Accessing Data With SQL

What is Apache Hive

hive.apache.org : data warehouse software that facilitates reading, writing, and managing large datasets residing in distributed storage using SQL

Wikipedia : a data warehouse infrastructure built on top of Hadoop for providing data summarization, query, and analysis

It provides :

- Tools to enable easy access to data via SQL, thus enabling data warehousing tasks such as extract/transform/load (ETL), reporting, and data analysis
- A mechanism to impose structure on a variety of data formats
- Access to files stored either directly in Apache HDFSTM or in other data storage systems such as Apache HBase
- Query execution via Apache TezTM, Apache SparkTM, or MapReduce

Hive is intended for data analysts who familiar with SQL

Hive was NOT

a relational database

- Hive uses a database to store metadata, but the data that Hive processes is stored in HDFS

designed for OLTP

- Hive runs on Hadoop
- Therefore, latency for Hive queries is generally high (even for small jobs)

suited for real-time queries and row-level updates

- Hive is best used for batch jobs over large sets of immutable data (such as web logs)

Schema On Read

- **At the time of loading into the hive, the data is not validated.**
  While in conventional database, the data is validated in accordance with a scheme that has been defined, if the data does not fit the scheme, it will be rejected.

- **Scheme is applied at the time of reading data**
  You can load the data before you know what to do with it, it offers you the ability to store structured, unlawful, and/or data that are not organized

Database

- A simply abstraction to group tables and other data unit
- To avoid naming conflicts for tables, views, partitions, columns, and so on
- Create database statement

```
CREATE DATABASE IF NOT EXISTS yava
```

- List all available database

```
SHOW DATABASES
```

- View a database description

```
DESC yava
```

- Use database

```
USE yava
```

Hands-On

Create Database

Create Database

1. Show connected user

```
SELECT logged_in_user()
```

2. Show all available databases

```
SHOW DATABASES
```

3. Create ***yava*** database

```
CREATE DATABASE IF NOT EXISTS yava
```

4. Show all available databases

```
SHOW DATABASES
```

5. View database information

```
DESC yava
```

Table

- A collection of related columns
- Can be filtered, projected, joined and unioned
- There are 2 types of tables

➔ **Managed tables**
managed by Hive by moving data into its warehouse directory
if tables are dropped, both data and metadata (schema) are deleted

➔ **External tables**
tables data will not be copied into hive warehouse directory
if tables are dropped only the schema from metastore will be deleted but not the data files from external location

Create Table

Example command to create a table :

```
CREATE TABLE IF NOT EXISTS youtube(
  video_id string,
  trending_date string,
  title string,
  channel_title string,
  category_id string,
  publish_time string,
  tags string,
  views string,
  likes string,
  dislikes string,
  comment_count string,
  thumbnail_link string,
  comments_disabled string,
  ratings_disabled string,
  video_error_or_removed string,
  description string
)
```

External Table

- Data stored in existing file in HDFS
- Tables and partition can be created
- File format must be in Hive-compatible format
- On dropping table, only the metadata drops

```
CREATE TABLE IF NOT EXISTS youtube(
  video_id string,
  trending_date string,
  title string
  )
ROW FORMAT serde 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
WITH SERDEPROPERTIES (
      "separatorChar" = ",",
      "escapeChar" = "\\"
)
STORED AS textfile
LOCATION '/user/yava/dataset/youtube'
TBLPROPERTIES (
      'skip.header.line.count' = '1')
```

Hands-On

Know Your Data

Download Dataset

1. Login to server using ***yava***
2. Download us_youtube_1K.csv dataset

```
# wget -P mydata https://www.dropbox.com/s/084rsqludghubkg/us_youtube_1K.csv
```

3. Verify dataset

```
# wget -P mydata https://www.dropbox.com/s/084rsqludghubkg/us_youtube_1K.csv
```

4. View content of file

```
# head mydata/us_youtube_1K.csv
```

5. Get number of line

```
# wc -l  mydata/us_youtube_1K.csv
```

Upload Dataset Into HDFS

1. Create new directory on HDFS dataset/youtube

```
# hdfs dfs -mkdir dataset/youtube
```

2. Upload us_youtube_1K.csv to HDFS

```
# hdfs dfs -put mydata/us_youtube_1K.csv dataset/youtube
```

3. Verify dataset on HDFS

```
# hdfs dfs -ls dataset/youtube
```

4. Count number of line

```
# hdfs dfs -cat dataset/youtube/us_youtube_1K.csv | wc -l
```

5. Change directory permission

```
# hdfs dfs -chmod -R 0777 dataset/youtube
```

Create Youtube External Table

1. Login to Hive
2. Show all available databases

```
SHOW DATABASES
```

3. Use yava database

```
USE yava
```

4. View all tables in yava database

```
SHOW tables
```

Create External Table

5. Create youtube external table

```
CREATE EXTERNAL TABLE IF NOT EXISTS youtube(
  video_id string,
  trending_date string,
  title string,
  channel_title string,
  category_id string,
  publish_time string,
  tags string,
  views string,
  likes string,
  dislikes string,
  comment_count string,
  thumbnail_link string,
  comments_disabled string,
  ratings_disabled string,
  video_error_or_removed string,
  description string
)
ROW FORMAT serde 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
WITH SERDEPROPERTIES (
       "separatorChar" = ",",
       "escapeChar" = "\\"
)
STORED AS textfile
LOCATION '/user/yava/dataset/youtube'
TBLPROPERTIES (
       'skip.header.line.count' = '1')
```

Create External Table

1. Show 5 sample of records

```
SELECT * FROM youtube LIMIT 5
```

2. Count number of records

```
SELECT count(1) num_rec FROM youtube
```

3. Compare with file on HDFS

Know Your Data

1. Get video_id length

```
SELECT LENGTH(video_id) as len_rec, count(*) as num_rec
FROM youtube
GROUP BY LENGTH(video_id)
```

2. Show records where the video_id length equal to 12

```
SELECT *
FROM youtube
WHERE LENGTH(video_id) = 12
LIMIT 5;
```

3. Show records where the video_id length is null

```
SELECT *
FROM youtube
WHERE LENGTH(video_id) is null
LIMIT 5;
```

Know Your Data

4. Show records where the video_id length equal to 11

```
SELECT *
FROM youtube
WHERE LENGTH(video_id) = 11
LIMIT 5;
```

5. Count number of record where the video_id length equal to 11

```
SELECT count(*) valid_rec
FROM <user>.youtube
WHERE LENGTH(video_id) = 11
```

Know Your Data

6. View some column length where video_id length is 11

```
SELECT LENGTH(video_id) len_video_id, LENGTH(trending_date) len_trending_date, LENGTH(category_id)
len_category_id, LENGTH(publish_time) len_publish_time, LENGTH(views) len_views,
      LENGTH(likes) len_likes, LENGTH(dislikes) len_dislike, count(*) num_rec
FROM youtube
GROUP BY LENGTH(video_id), LENGTH(trending_date), LENGTH(category_id), LENGTH(publish_time), LENGTH(views),
LENGTH(likes), LENGTH(dislikes)
HAVING len_video_id = 11;
```

7. Show records where video_id length is 11 and views length is 6

```
SELECT *
FROM youtube
WHERE LENGTH(video_id) = 11 AND LENGTH(views) = 6
LIMIT 5;
```

Create New Table for Valid Data

1. Create stg_youtube table

```
CREATE TABLE IF NOT EXISTS stg_youtube
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t'
STORED AS textfile
AS SELECT
  video_id,
  CAST(concat( substr(trending_date,1,2),
       substr(trending_date,7,2),
       substr(trending_date,4,2))
       as int) trending_date,
  title,
  channel_title,
  CAST(category_id as int),
  publish_time,
  tags,
  CAST(views as int),
  CAST(likes as int),
  CAST(dislikes as int),
  CAST(comment_count as int)
FROM youtube
WHERE LENGTH(video_id) = 11 AND LENGTH(trending_date) = 8;
```

Create New Table for Valid Data

2. View stg_youtube table description

```
describe stg_youtube
```

3. Count number of records

```
select count(*) num_rec from stg_youtube
```

4. How many days in dataset?

```
select count(1) num_days
from(
 select
      trending_date
 from stg_youtube
 group by trending_date
) a;
```

Create New Table for Valid Data

5. Get number of records for each day

```
select
      trending_date,
      count(*) num_rec
from stg_youtube
group by trending_date
order by trending_date;
```

6. Show records

```
select * from stg_youtube limit 10;
```

Chapter 08

HBase - NoSQL Database

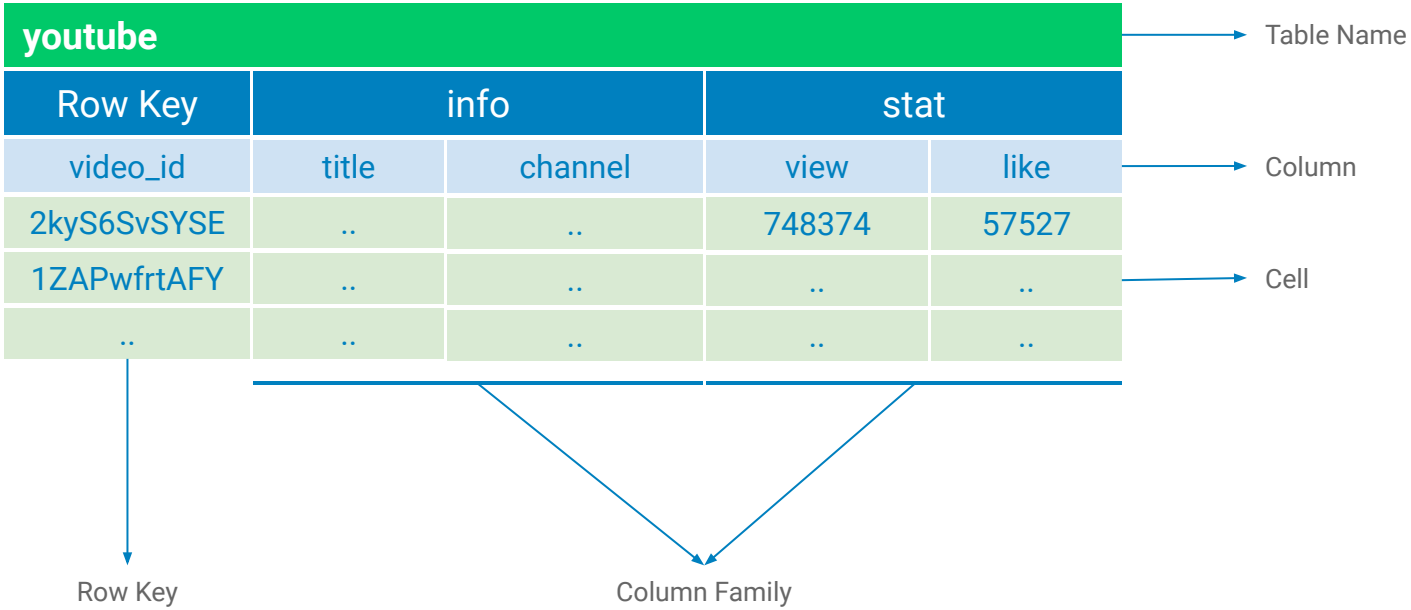
What is HBase?

- An open-source, distributed, versioned, non-relational database
- Use Apache HBase when you need random, realtime read/write access to your Big Data
- Modeled after Google's Bigtable:

  A Distributed Storage System for Structured Data by Chang et al

- Build on top of HDFS
- Provide fast lookups for larger tables
- Technically speaking, HBase is really more a "Data Store" than "Database" because it lacks many of the features you find in an RDBMS

Data Model


youtube
Row Key
info
stat
video_id
title
channel
view
like
2kyS6SvSYSE
..
..
748374
57527
1ZAPwfrtAFY
..
..
..
..
..
..
..
..
..
Table Name
Column
Cell
Row Key
Column Family

How To Code ?

1. Start HBase shell

```
# hbase shell
```

2. Show all available namespace

```
hbase(main):001:0> list_namespace
```

```
NAMESPACE
default
hbase
```

3. Create ***yava*** namespace

```
hbase(main):010:0> create_namespace 'yava'
```

4. Create ***youtube*** table, with ***info*** and ***stat*** column family

```
hbase(main):010:0> create_namespace 'yava'
```

How To Code ?

5. Show all table

```
hbase(main):001:0> list
```

```
TABLE
Sensors
customer
yava:youtube
```

6. Show all table in ***yava*** namespace

```
hbase(main):001:0> list_namespace_tables 'yava'
```

```
TABLE
youtube
```

How To Code ?

7. Show all table

```
hbase(main):019:0> put 'user20:youtube', '2kyS6SvSYSE', 'info:title', 'WE WANT TO TALK ABOUT ...'
hbase(main):020:0> put 'user20:youtube', '2kyS6SvSYSE', 'info:channel', 'CaseyNeistat'
hbase(main):021:0> put 'user20:youtube', '2kyS6SvSYSE', 'stat:view', '748374'
hbase(main):022:0> put 'user20:youtube', '2kyS6SvSYSE', 'stat:like', '57527'
hbase(main):023:0> put 'user20:youtube', '2kyS6SvSYSE', 'stat:dislike', '2966'
hbase(main):024:0> put 'user20:youtube', '2kyS6SvSYSE', 'stat:comment', '15954'
```

8. Show all table in ***yava*** namespace

```
hbase(main):001:0> scan 'yava:youtube'
```

```
ROW                 COLUMN+CELL
 2kyS6SvSYSE        column=info:channel, timestamp=1584008519807, value=CaseyNeistat
 2kyS6SvSYSE        column=info:title, timestamp=1584008512600, value=WE WANT TO TALK ABOUT ..
 2kyS6SvSYSE        column=stat:comment, timestamp=1584008550568, value=15954
 2kyS6SvSYSE        column=stat:dislike, timestamp=1584008542611, value=2966
 2kyS6SvSYSE        column=stat:like, timestamp=1584008535453, value=57527
 2kyS6SvSYSE        column=stat:view, timestamp=1584008529649, value=748374
```

How To Code ?

9. Get a row

```
hbase(main):001:0> get 'yava:youtube', '2kyS6SvSYSE'
```

```
COLUMN              CELL
 info:channel       timestamp=1584008519807, value=CaseyNeistat
 info:title         timestamp=1584008512600, value=WE WANT TO TALK ABOUT OUR MARRIAGE
 stat:comment       timestamp=1584008550568, value=15954
 stat:dislike       timestamp=1584008542611, value=2966
 stat:like          timestamp=1584008535453, value=57527
 stat:view          timestamp=1584008529649, value=748374
```

Column Family Versioning

1. Table description

```
hbase(main):001:0> desc 'yava:youtube'
```

```
Table yava:youtube is ENABLED
yava:youtube
COLUMN FAMILIES DESCRIPTION
{NAME => 'info', VERSIONS => '1', EVICT_BLOCKS_ON_CLOSE => 'false', NEW_VERSION_BEHAVIOR => 'false',
……
```

2. Alter Column Family version number

```
hbase(main):001:0> alter 'yava:youtube', {NAME => 'info',VERSIONS => '4'}
```

3. View table description

```
hbase(main):001:0> desc 'yava:youtube'
```

Column Family Versioning

4. Update a cell

```
hbase(main):001:0> put 'yava:youtube', '2kyS6SvSYSE', 'info:channel', 'new channel'
```

5. View historical a cell

```
hbase(main):001:0> get 'yava:youtube', '2kyS6SvSYSE', {COLUMN => 'info:channel', VERSIONS => 2}
```

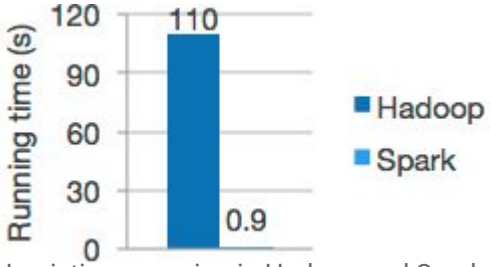
```
COLUMN                    CELL
 info:channel             timestamp=1584010398282, value=2kyS6SvSYSE
 info:channel             timestamp=1584010193742, value=new channel
```

Chapter 09

Apache Spark - Analytics Engine

What is Apache Spark?

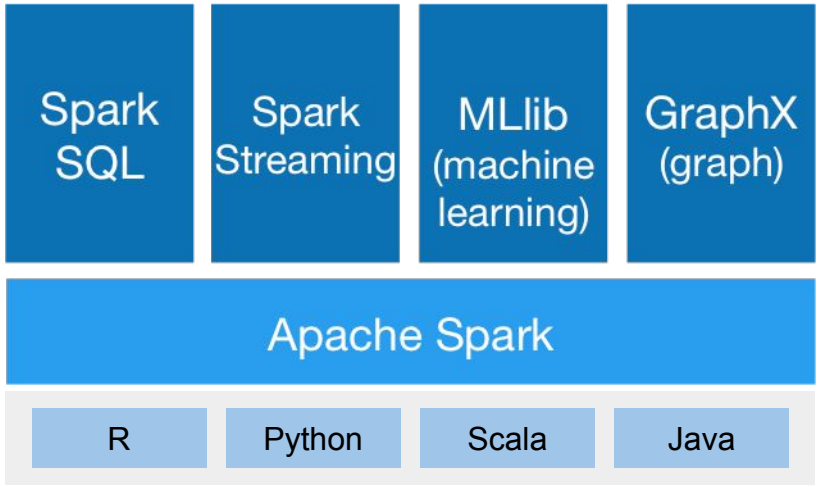
- Lightning-fast unified analytics engine for large-scale data processing
- The main feature of Spark is its in-memory cluster computing that increases the processing speed of an application
- Run workloads 100x faster



Logistic regression in Hadoop and Spark
img source: spark.apache.org

- Spark is one of Hadoop's sub project developed in 2009
- in UC Berkeley's AMPLab by Matei Zaharia
- Apache Spark is built by a wide set of developers from over 300 companies.
- Since 2009, more than 1200 developers have contributed to Spark

Component



1. **Spark Core**
   Provide core component for in memory distributed data processing

3. **SparkSQL**
   Spark SQL lets you query structured data inside Spark programs, using either SQL or a familiar DataFrame API. Usable in Java, Scala, Python and R

4. **Spark Streaming**
   Spark Streaming brings Apache Spark's language-integrated API to stream processing, letting you write streaming jobs the same way you write batch jobs

5. **MLlib**
   Provide scalable machine learning library and high-quality algorithms, 100x faster than MapReduce

6. **GraphX**
   a library added in Spark 0.9 that provides an API for manipulating graphs
   (e.g., a social network's friend graph)

Why Apache Spark?

- Its fast
- Integrate with Hadoop and its ecosystem and can read existing data
- Provide high level API : Scala, Java, Python, R
- Most of machine learning algorithm are iterative
- Can be implemented in standalone mode, Amazon EC2, Mesos and YARN

Let Use Python

Why Python

- ➔ It's a lot simpler, and this is just an overview
- ➔ Don't need to compile anything, deal with JAR's, dependencies, etc

But

- ➔ Spark itself is written in Scala
- ➔ Scala's functional programming model is a good fit for distributed processing
- ➔ Give fast performance (Scala compiles to Java bytecode)
- ➔ Less code & boilerplate stuff thanJava
- ➔ Python is slow

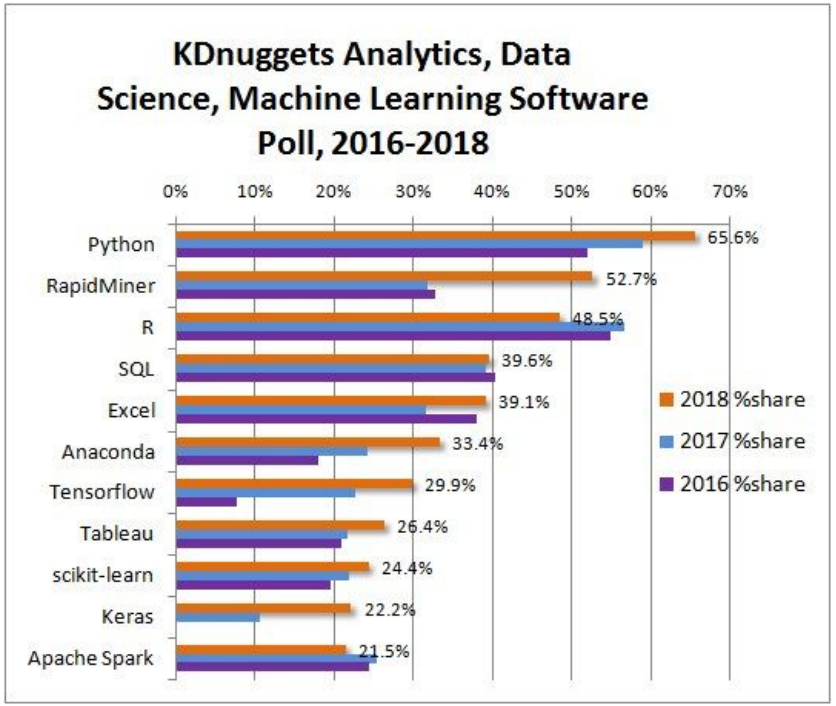
Let Use Python

- Python code

```
nums = sc.parallelize([1, 2, 3, 4])
squared = nums.map(lambda x: x * x).collect()
```

- Scala

```
val nums = sc.parallelize(List(1, 2, 3, 4))
Val squared = nums.map(x => x * x).collect()
```



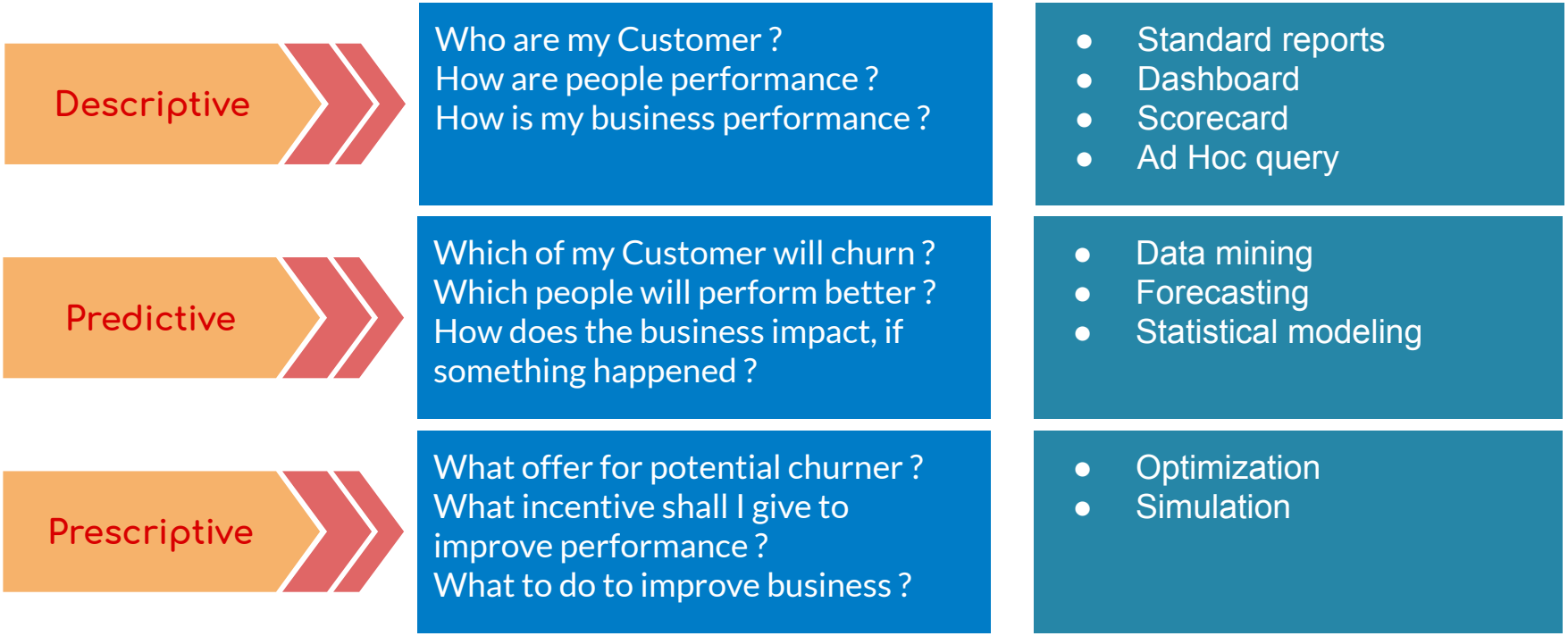
Hands-On

Answering Business Question

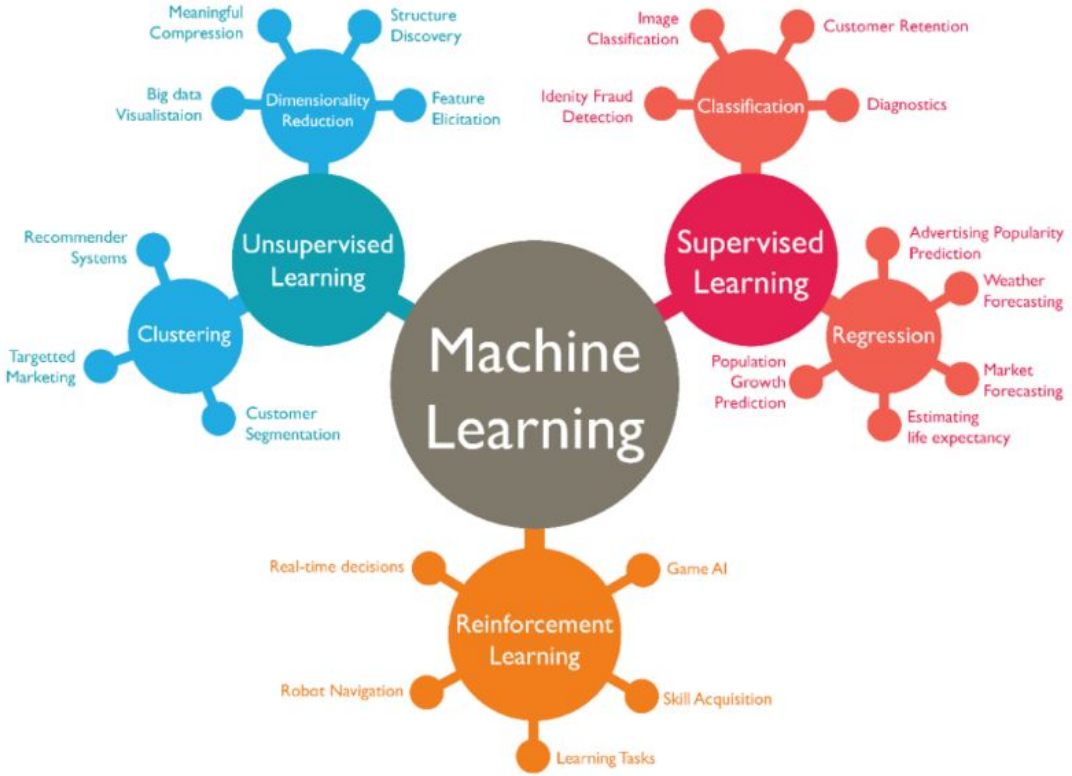
Chapter 10

Artificial Intelligent

Data Analytics Method



Machine Learning



Hands-On

Machine Learning With PySpark

Deep Learning

Artificial Intelligent

the science of getting computers to act in specific ways without explicitly programming them to do so.

Machine Learning

the science of getting computers to act in specific ways without explicitly programming them to do so.
There are a number of machine learning methods or algorithms that can be applied to almost any data problem, i.e

- Regression
- K-Means
- Decision tree
- Random Forest
- etc

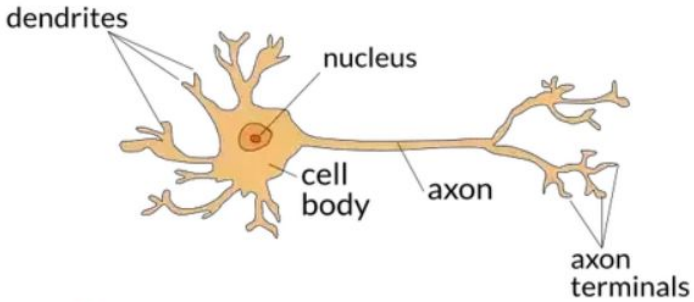
Deep Learning

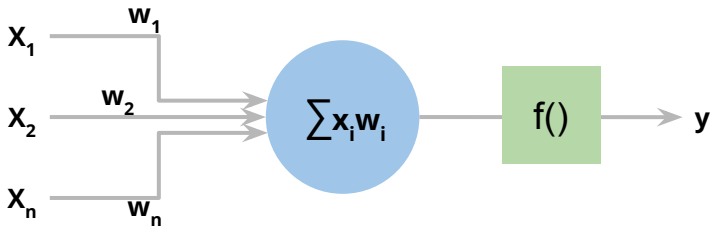
a form of machine learning that is inspired by the structure of the human brain and is particularly effective in feature detection.
Open source framework :

- Theano
- Tensor Flow
- Torch
- Caffe
- MXNet
- etc

What is Deep Learning?

- Machine learning algorithms based on learning multiple levels (i.e deep) of representation/ abstraction (1)
- Learning algorithms derive meaning out of data by using a hierarchy of multiple layers of units (neurons)
- Each neuron/node computes a weighted sum of its inputs and the weighted sum is passed through a nonlinear function, each layer transforms input data in more and more abstract representations
- Learning = find optimal weights from data



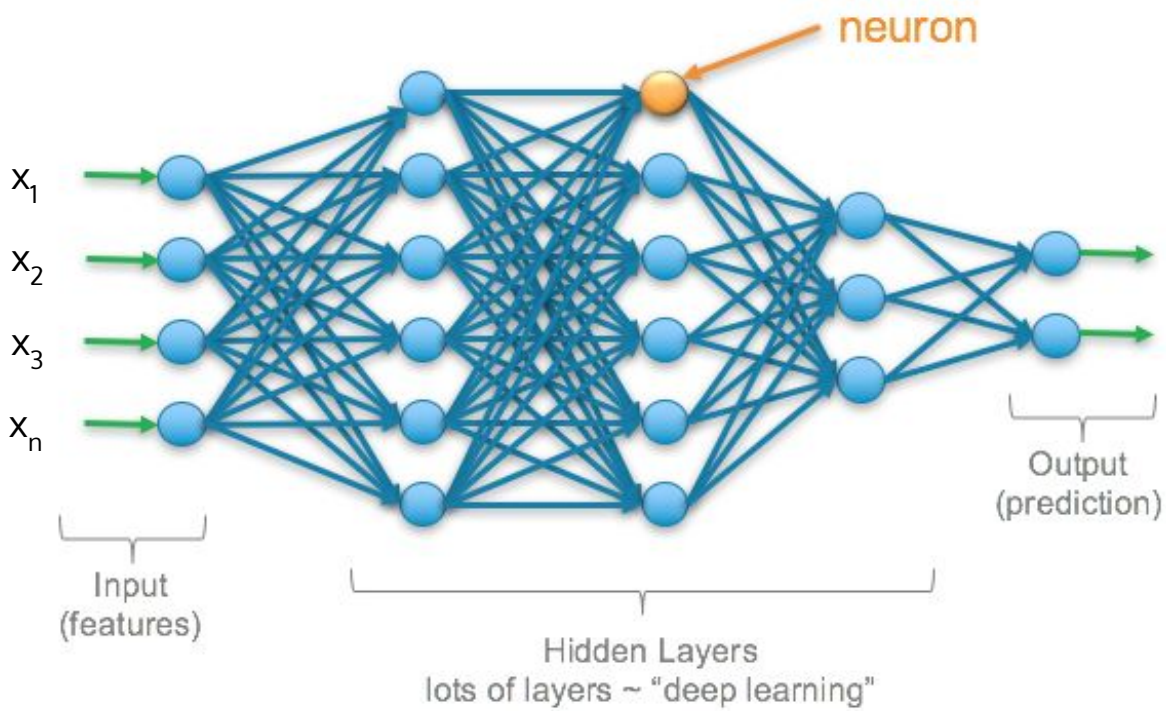


Why Now?

- Exponential data growth (and the ability to Process Structured & Unstructured data)
- Faster & open distributed systems (Hadoop, Spark, TensorFlow, ...)
- Faster machines and multicore CPU/GPUs
- New and better models, algorithms, ideas:
  - Better, more flexible learning of intermediate representations
  - Effective end-to-end joint system learning
  - Effective learning methods for using contexts and transferring between tasks

"The analogy to deep learning is that the ***rocket engine*** is the deep learning models and the fuel is the ***huge amounts of data*** we can feed to these algorithms." - Andrew Ng

Multilayer Perceptron



$y_1 = P(cat|x)$

$y_2 = P(dog|x)$

GAN Implementation

Figure 5: 1024 × 1024 images generated using the CELEBA-HQ dataset. See Appendix F for a larger set of results, and the accompanying video for latent space interpolations.

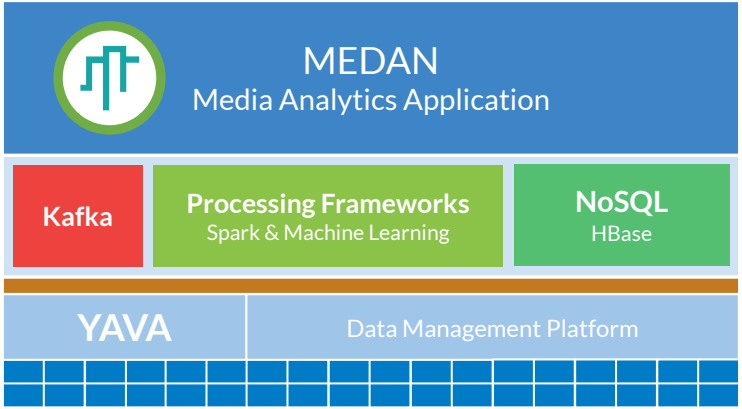
Paper :
Progressive Growing Of GANS For Improved Quality, Stability, And Variation - NVIDIA

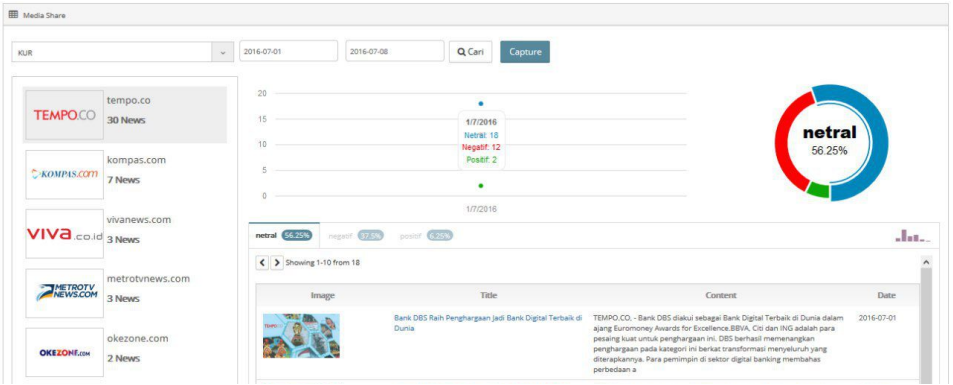
https://arxiv.org/pdf/1710.10196.pdf

Chapter 11

Big Data Applications

Media Analytics

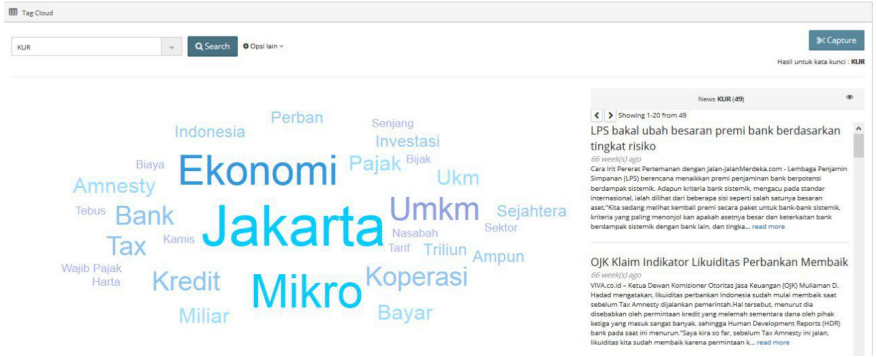


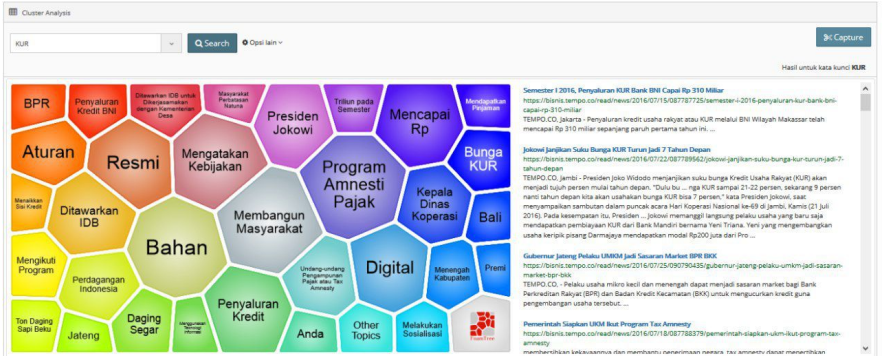


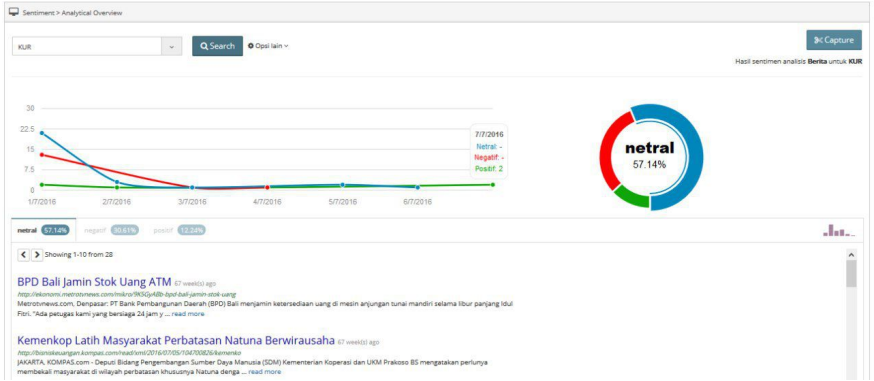
Capabilities :

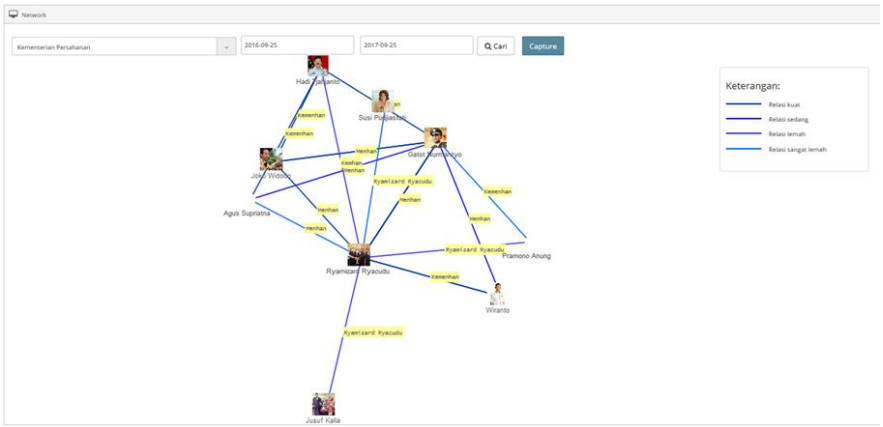
- Crawling Media Online, Forums, SocMed
- Sentiment Analytics, Tag CLouds, Cluster Analytics, Geolocation, Network Analytics, Top Person, etc

Media Analytics

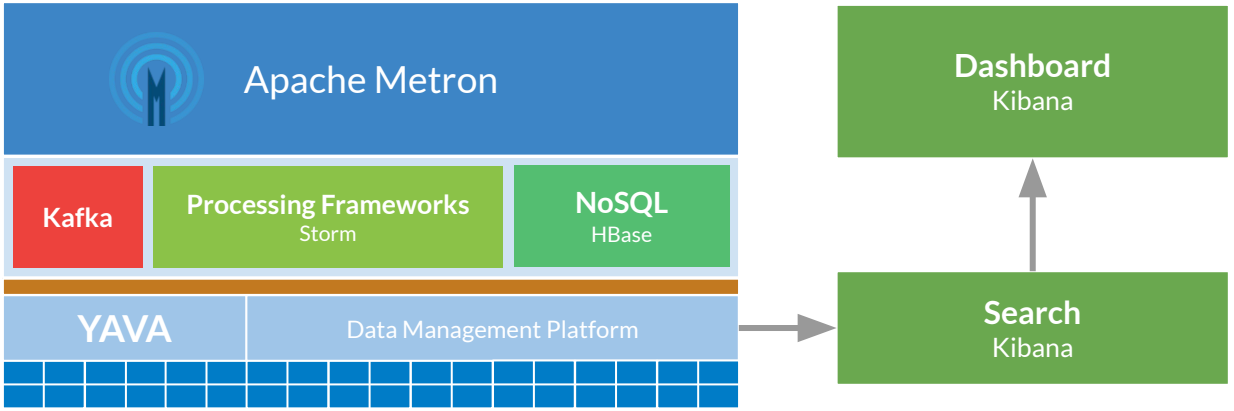






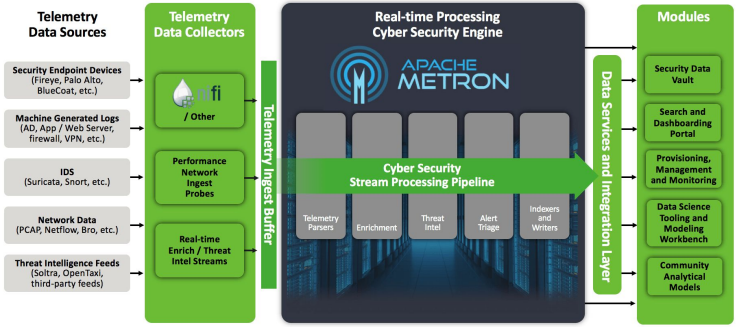


Network Security Analyzer

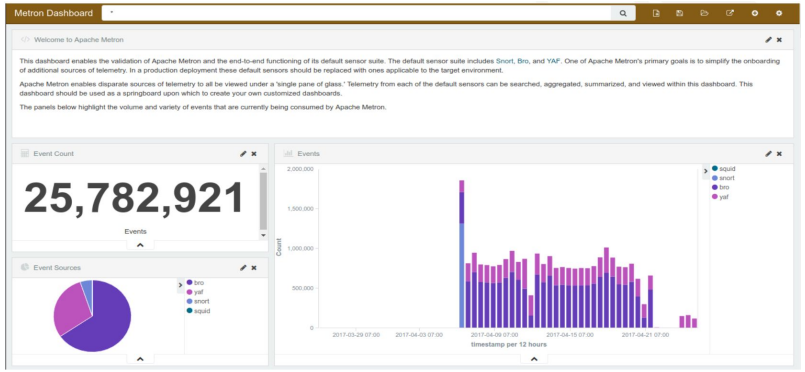


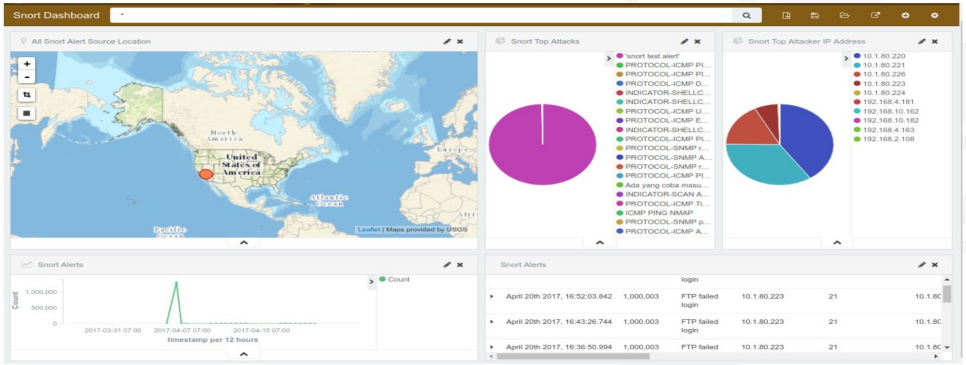
Capabilities :

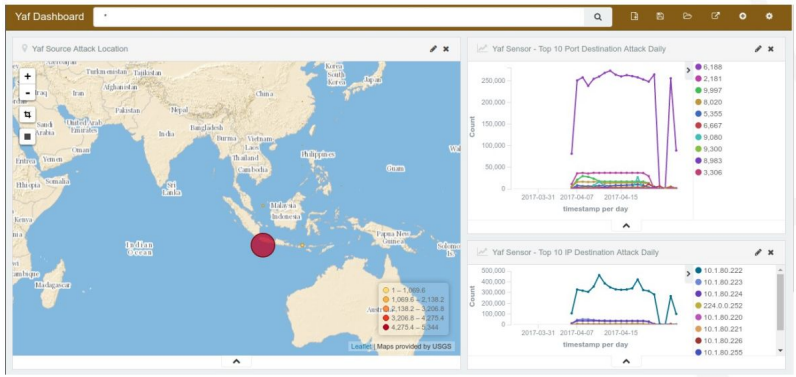
- Security Data Lake
- Pluggable Framework
- Threat Detection Platform
- Incident Response Application

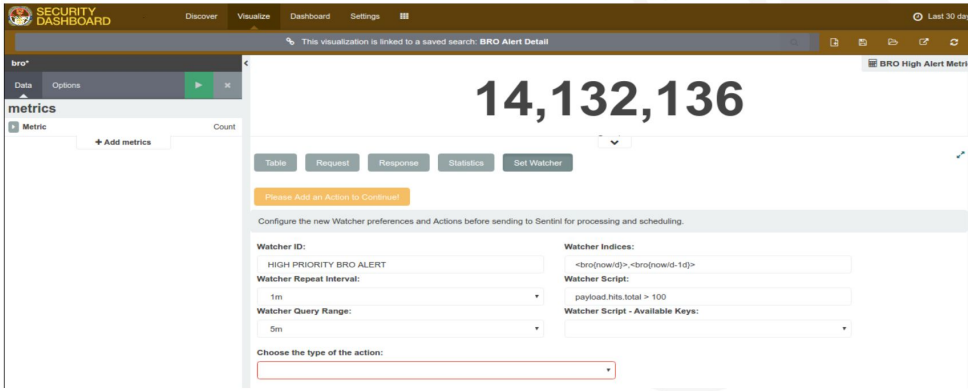


Network Security Analyzer

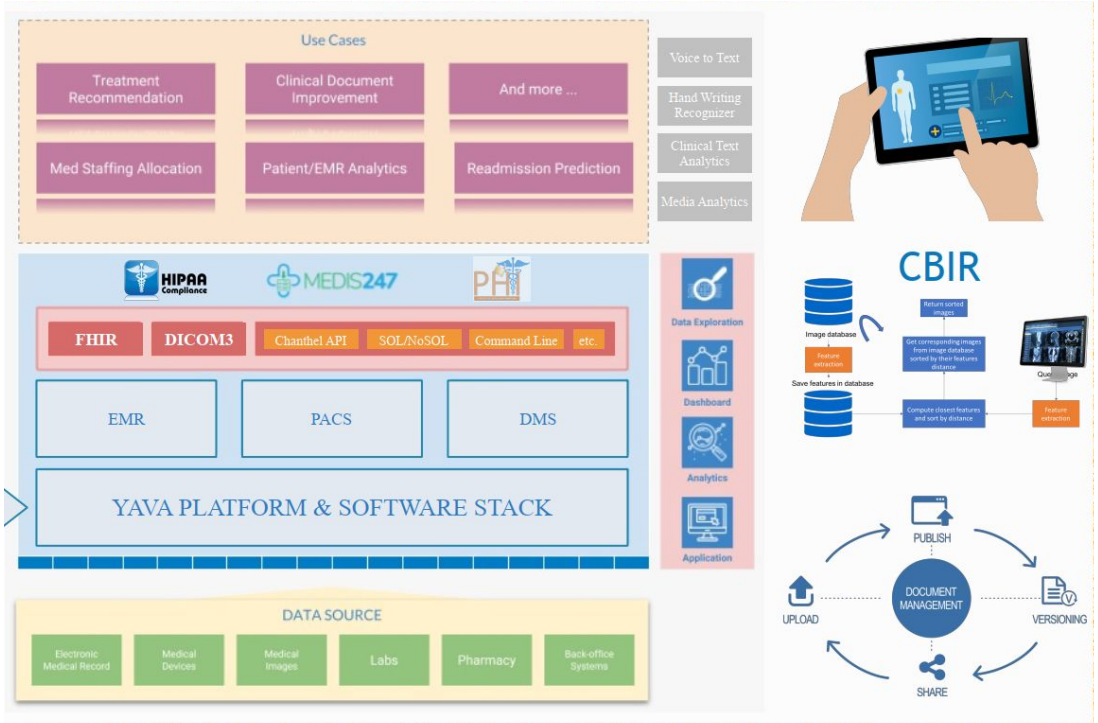








Medis247



- Analytics Dashboard
- Electronic Medical Records - HL7
- Integration with ICU Devices
- Automatic Speech Recognition

Important Links

1. Notebook in this course

   https://github.com/project303/DEV122---Yava-Essentials

2. YavaCE Cookbooks

   https://github.com/project303/YavaCE-Cookbook

3. Yava Community Edition

   https://yava.labs247.id/download_box/

Development team:
Sigit Prasetyo
M. Urfah

THE REAL TRAINING BEGINS WHEN THE CLASS ENDS

- DR. Billy Kueek