NoSQL

Introduction to NoSQL (MongoDB and Elastic)

By :

Mehdi Habibzadeh (@NimaHM1980)

Hossein Shemshadi (@HosseinShemshadi)

July 2017

Outlines (August 2017) :

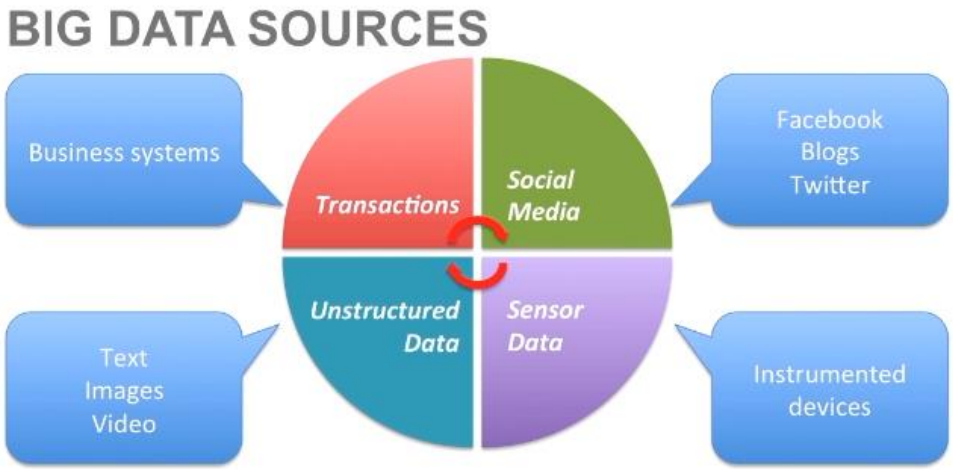
- **Big Data and Challenges**
  - Review and Trends
  - Map-Reduce on Large Clusters
  - Hadoop Framework Programming
  - Spark and Flink Frameworks
  - Big Data and Cloud Computing
  - Big Data and NoSQL (Frameworks and Security Issue)
    - MongoDB
    - Elasticsearch
    - Titan
  - Big Data in the Real World

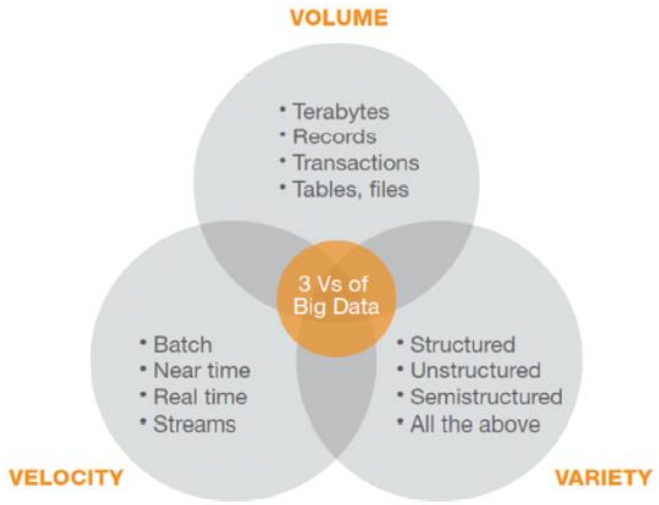
Big Data and Challenges

Sources and Massive Information

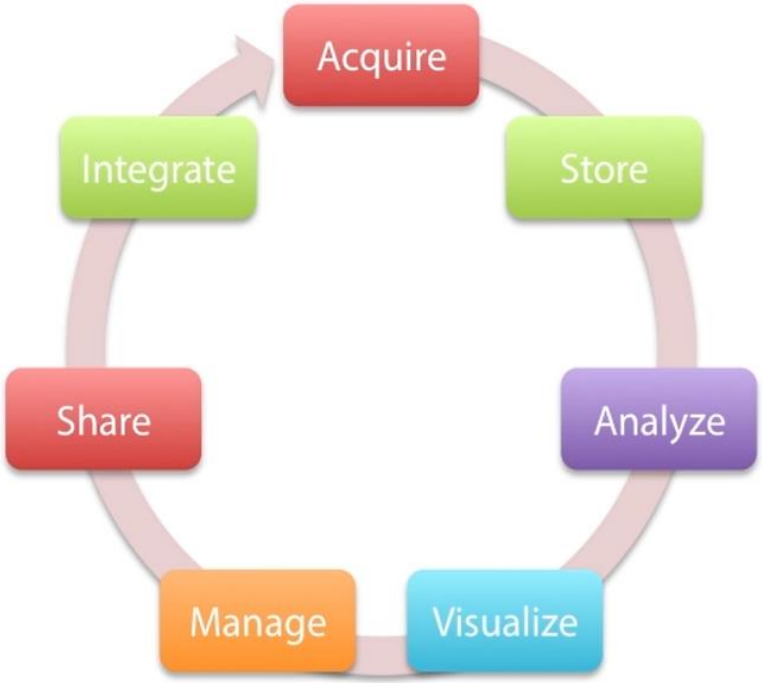
Characteristics and Trends

- The year 2015 was a big jump in the world of big data.
  - » Adoption of technologies, associated with unstructured data
  - » Ref : http://www.tableau.com/top-8-trends-big-data-2017?

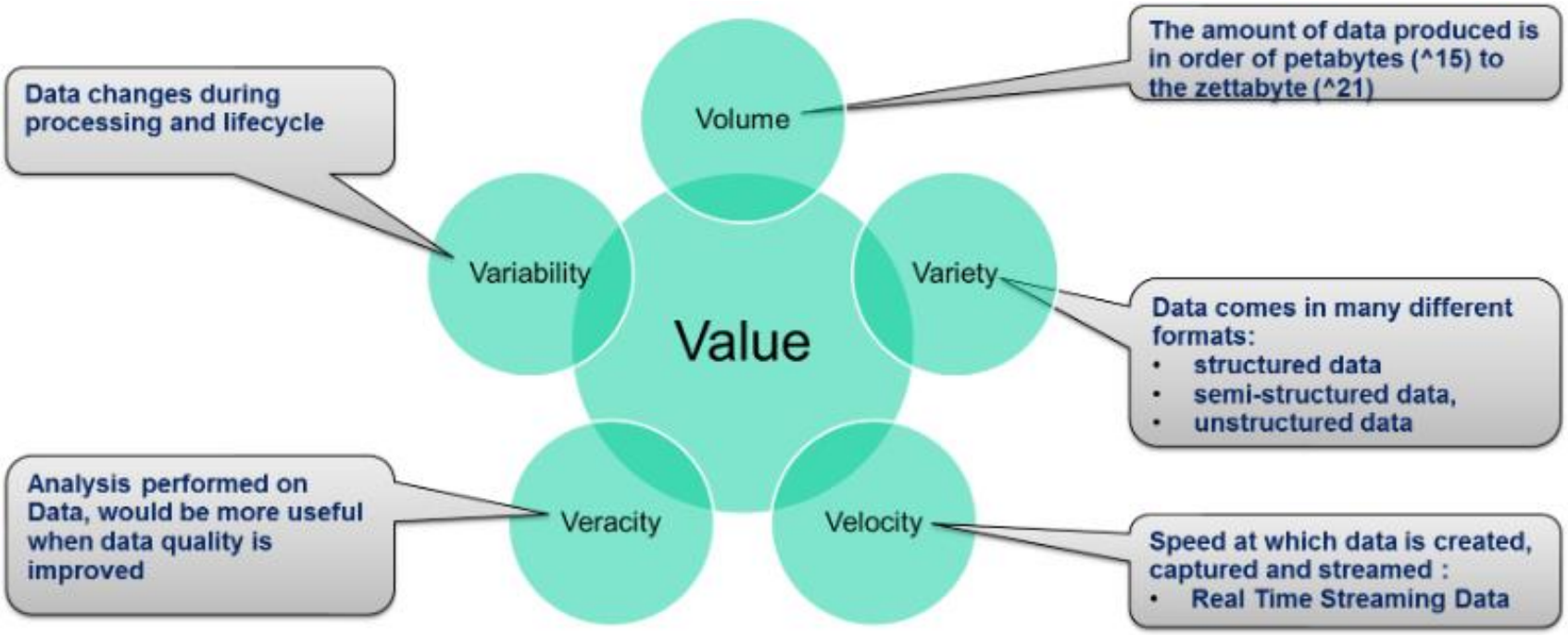




Big Data and Challenges (Cont.)



Big Data and Challenges (Cont.)

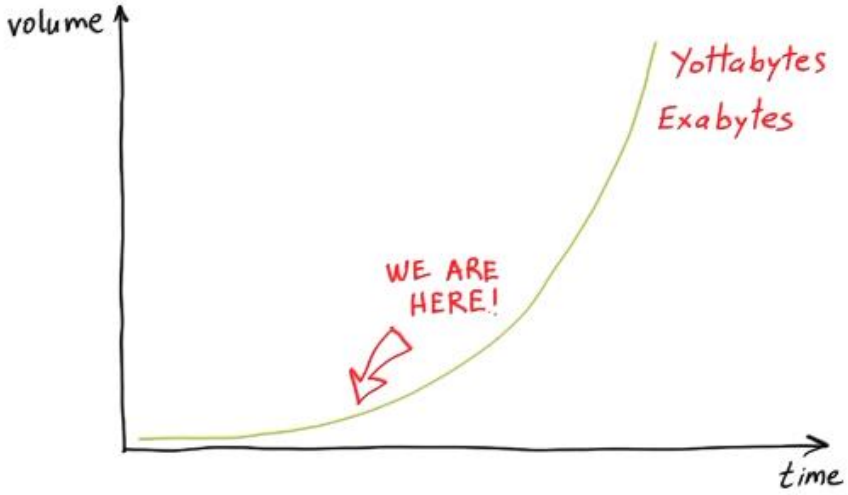


Big Data and Challenges (Cont.)

Value		Clinically relevant data Longitudinal studies
Volume		High-throughput technologies Continuous monitoring of vital signs
Velocity		High-speed processing for fast clinical decision support Increasing data generation rate by the health infrastructure
Variety		Heterogeneous and unstructured data sources Differences in frequencies and taxonomies
Veracity		Data quality is unreliable Data coming from uncontrolled environments
Variability		Seasonal health effects and disease evolution Non-deterministic models of illness and health

Big Data and Challenges (Cont.)

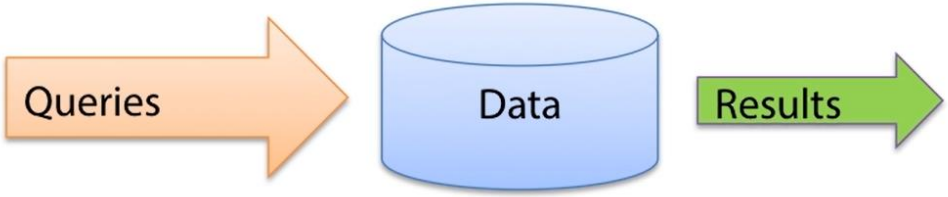
- Phone: AT&T 20TB phone call database, wireless tracking
- Consumer: WalMart 70TB database, buying patterns
- WEB: Web crawl of 200M pages and 2000M links, Akamai stores 7 billion clicks per day
- Geography: NASA satellites generate 1.2TB per day



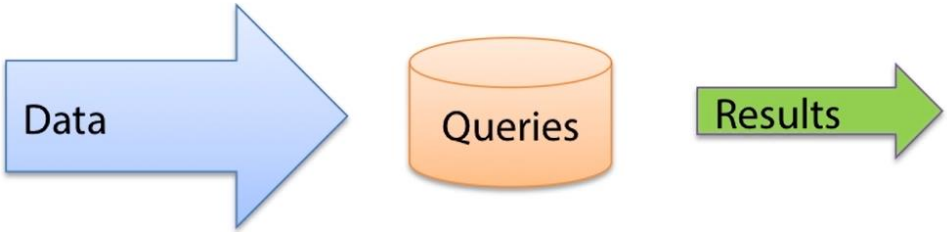
Big Data Retrieval Algorithms

- **Streaming**
  - Online Data Management
  - Adapt to arbitrary and unstructured Input Data
  - Real-Time Analytical Processing (RTAP)

- Conventional processing: **static data**

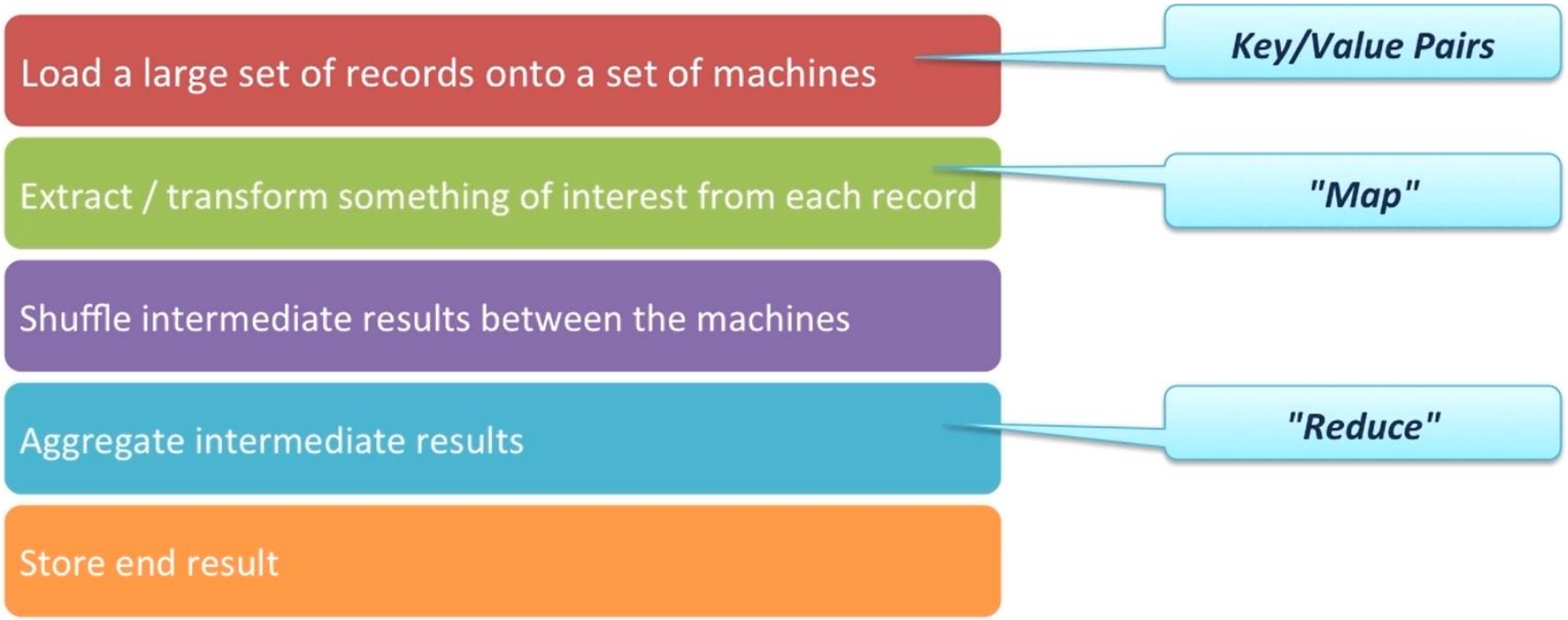


- Real-time processing: **streaming data**



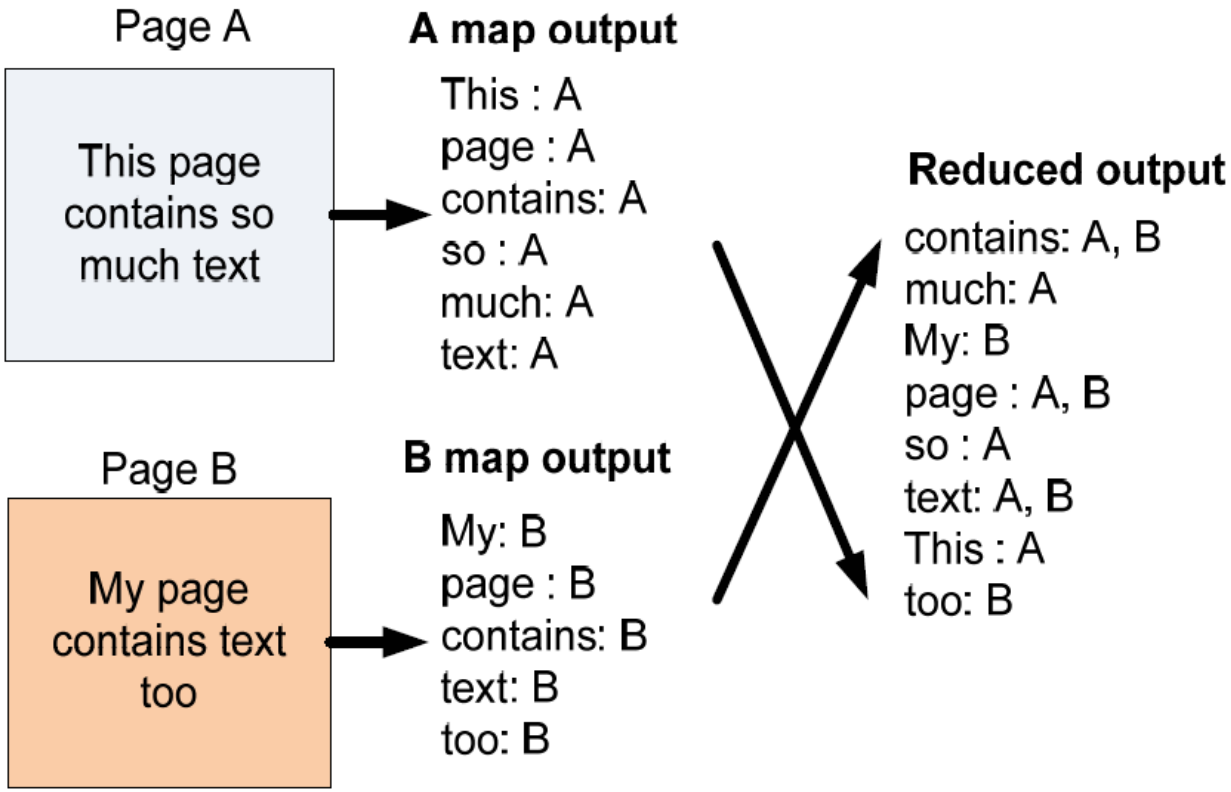
Map-Reduce on Large Clusters

- **Motivation and Demand:**
  - Tend to be very short, code-wise
  - Represent a data flow

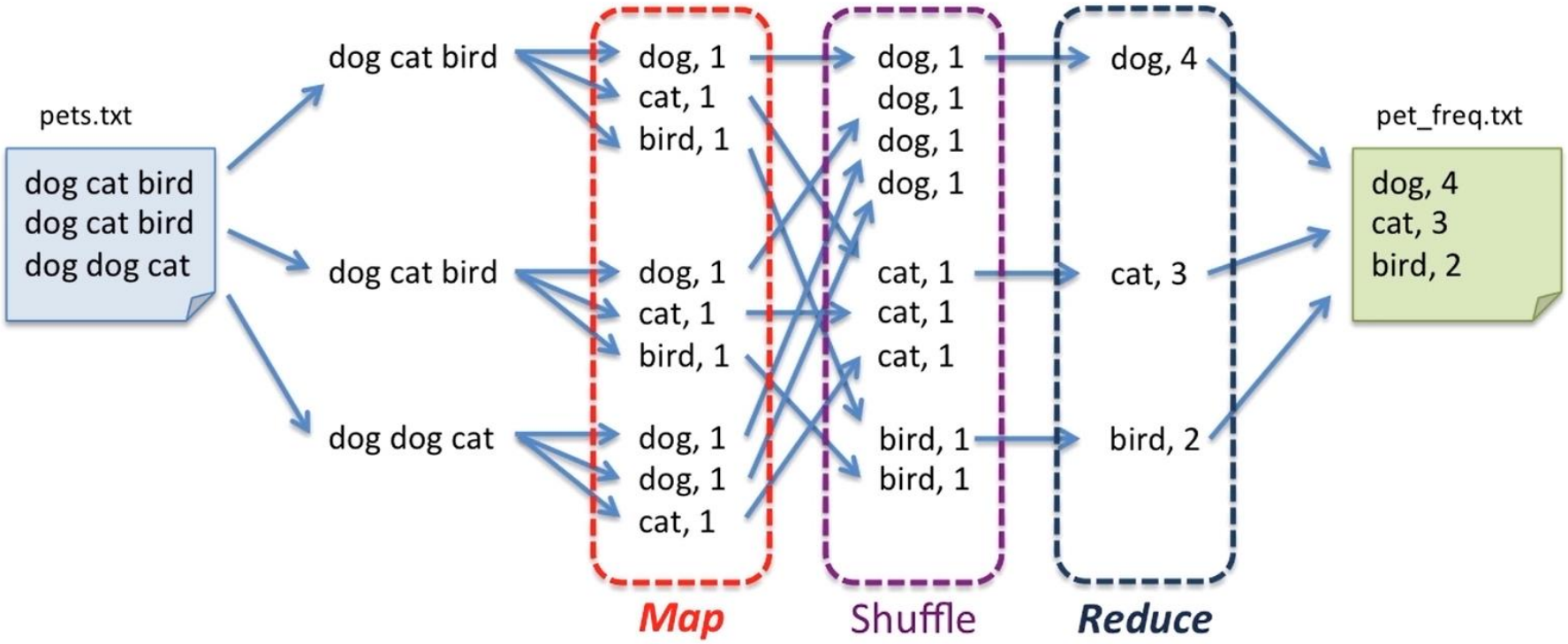


Map-Reduce (Cont.)

Index: Data Flow

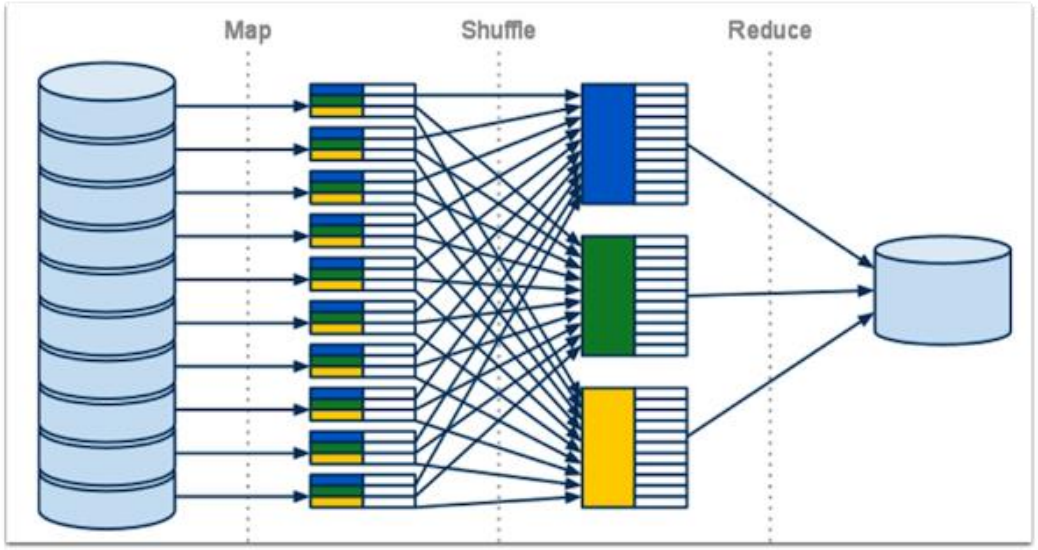


Map-Reduce (Cont.)



Map-Reduce (Cont.)

- **Each step has one Map phase and one Reduce phase**
  - Convert any into MapReduce pattern
- **Great solution for one-pass computations**
  - Not very efficient for Multi-pass computations and algorithms



Hadoop Framework

- **Features :**
  - Open Source Framework for Processing Large Data
  - Work on Cheap and Unreliable Clusters
  - Known in Companies who deal with Big Data Applications
  - Compatible with Java, Python and Scala

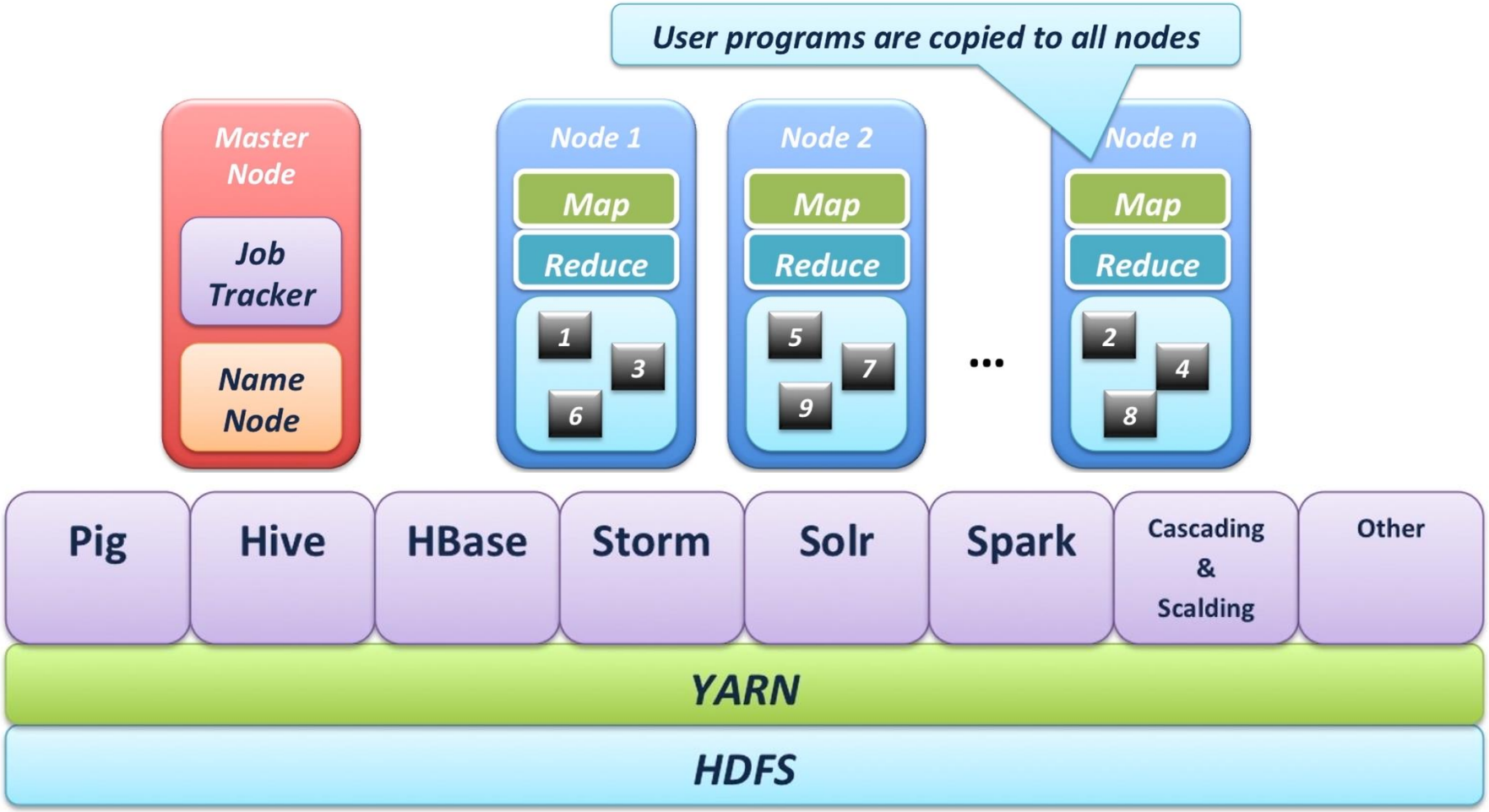
Programmers	Framework
• Map • Reduce	• Deals with fault tolerance • Assign workers to map and reduce tasks • Moves processes to data • Shuffles and sorts intermediate data • Deals with errors

Hadoop Framework (Cont.)

- **MapReduce Framework**
  - Assign work for different nodes
- **Hadoop Distributed File System (HDFS)**
  - Primary storage system used by Hadoop applications.
  - Copies each piece of data and distributes to individual nodes
    - Name Node (Meta Data) and Data Nodes (File Blocks)
    - Redundant information (Three times by default)
    - Machines in a given cluster are cheap and unreliable
    - Decreases the risk of catastrophic failure
      - » Even in the event that numerous nodes fail
  - Links together the file systems on different nodes to make an integrated big file system (Parallel Processing)

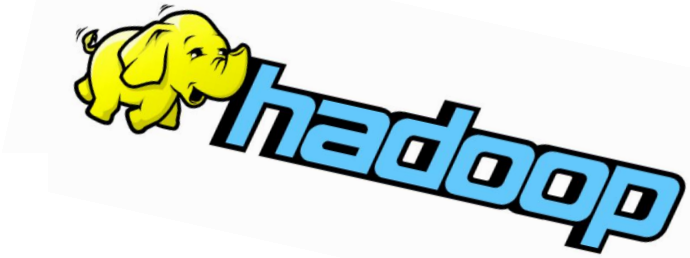
Hadoop Framework (Cont.)

- **Hadoop V.2 :** Hadoop NextGen MapReduce **(YARN)**



Hadoop Framework (Cont.)

- **Hadoop Programming**
  - Java
    - Full control of MapReduce , Cascading (Open Java Library)
  - Python , Scala, Ruby
- **Data Retrieval / Query Language**
  - Hive
    - SQL- Like Language
  - Pig
    - Data Flow Language (Simple and Out of Small Steps)
  - Scalding
    - Library built on top of Scala (Elegant Model)



Big Data Programming

- R – Java- Python and Scala (Commonly Used)
- Three References : (Recommended to Read)
  - https://www.linkit.nl/knowledge-base/177/4_most_used_languages_in_big_data_projects_Java
  - https://www.linkit.nl/knowledge-base/226/4_most_used_languages_in_big_data_projects_R
  - https://www.linkit.nl/eng/knowledge-base/196/4_most_used_languages_in_big_data_projects_Python

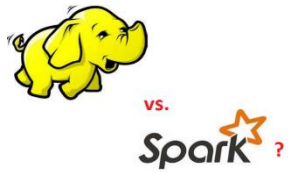
Hadoop Framework (Cont.)

Table 1
Platforms & tools for big data analytics in healthcare

Platform/Tool	Description
The Hadoop Distributed File System (HDFS)	HDFS enables the underlying storage for the Hadoop cluster. It divides the data into smaller parts and distributes it across the various servers/nodes.
MapReduce	MapReduce provides the interface for the distribution of sub-tasks and the gathering of outputs. When tasks are executed, MapReduce tracks the processing of each server/node.
PIG and PIG Latin (Pig and PigLatin)	Pig programming language is configured to assimilate all types of data (structured/unstructured, etc.). It is comprised of two key modules: the language itself, called PigLatin, and the runtime version in which the PigLatin code is executed.
Hive	Hive is a runtime Hadoop support architecture that leverages Structure Query Language (SQL) with the Hadoop platform. It permits SQL programmers to develop Hive Query Language (HQL) statements akin to typical SQL statements.
Jaql	Jaql is a functional, declarative query language designed to process large data sets. To facilitate parallel processing, Jaql converts "'high-level' queries into 'low-level' queries" consisting of MapReduce tasks.
Zookeeper	Zookeeper allows a centralized infrastructure with various services, providing synchronization across a cluster of servers. Big data analytics applications utilize these services to coordinate parallel processing across big clusters.
HBase	HBase is a column-oriented database management system that sits on top of HDFS. It uses a non-SQL approach.
Cassandra	Cassandra is also a distributed database system. It is designated as a top-level project modeled to handle big data distributed across many utility servers. It also provides reliable service with no particular point of failure (http://en.wikipedia.org/wiki/Apache_Cassandra) and it is a NoSQL system.
Oozie	Oozie, an open source project, streamlines the workflow and coordination among the tasks.
Lucene	The Lucene project is used widely for text analytics/searches and has been incorporated into several open source projects. Its scope includes full text indexing and library search for use within a Java application.
Avro	Avro facilitates data serialization services. Versioning and version control are additional useful features.
Mahout	Mahout is yet another Apache project whose goal is to generate free applications of distributed and scalable machine learning algorithms that support big data analytics on the Hadoop platform.

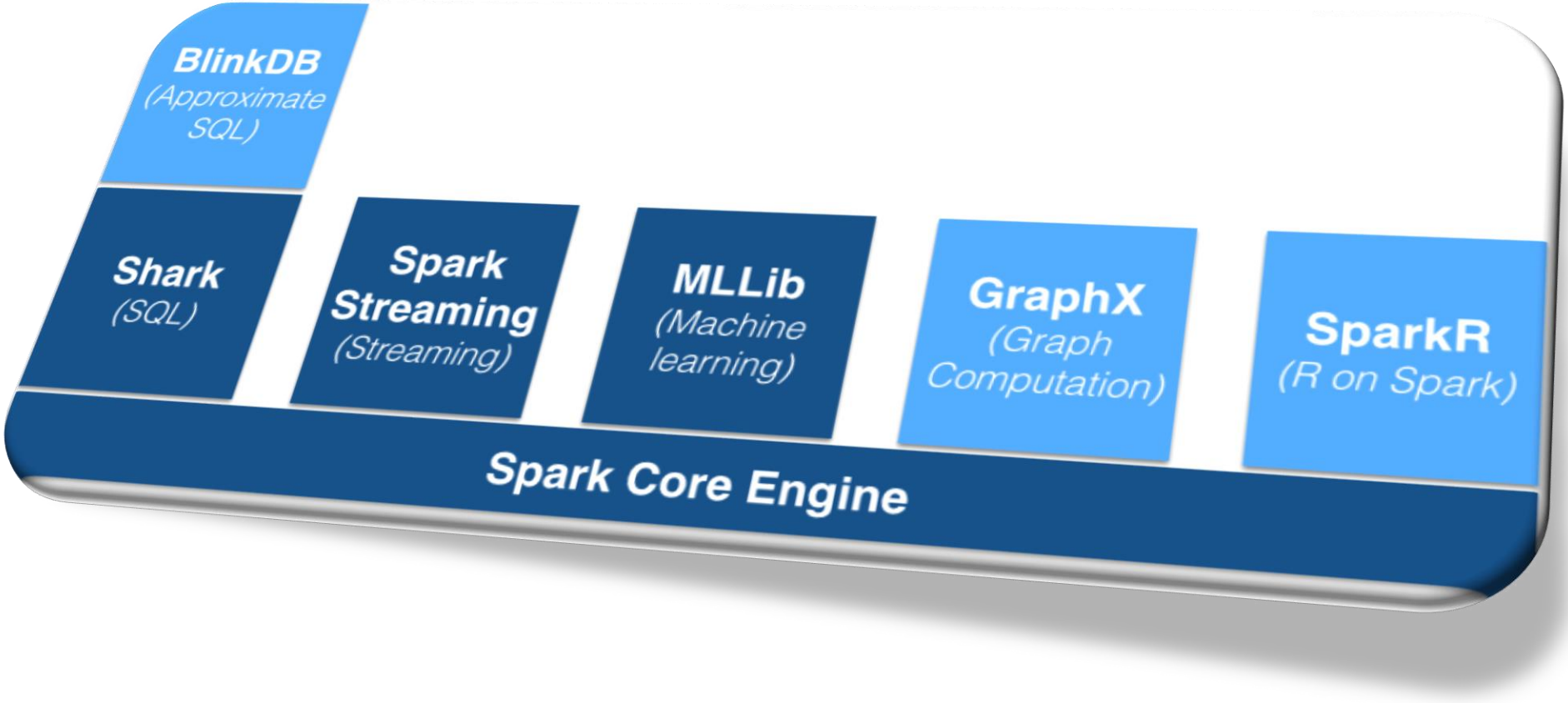
Apache Spark Framework

- **Spark Features (More than Distributed Processing)**
  - Ease of use, and sophisticated analytics
  - In-memory data storage and near real-time processing
  - Holds intermediate results in memory
  - Store as much as data in memory and then goes to disk
- **Spark vs Hadoop**



  - On top of existing HDFS
  - Data sets that are diverse in nature (Text, Videos, …)
  - Variety in source of data (Batch v. real-time streaming data).
  - 100 times faster in memory, 10 times faster when running on disk.

Apache Spark Framework (Cont.)



Apache Spark Framework (Cont.)

- **Compatible with Java, Scala and Python**
- **Perform Data Analytics and Machine Learning**
  - SQL Queries, Streaming Data
  - Machine Learning and Graph Data Processing
    - Spark MLlib, Spark's Machine Learning library
- **Spark and data stored in a Cassandra database**
  - **(Case Study)**

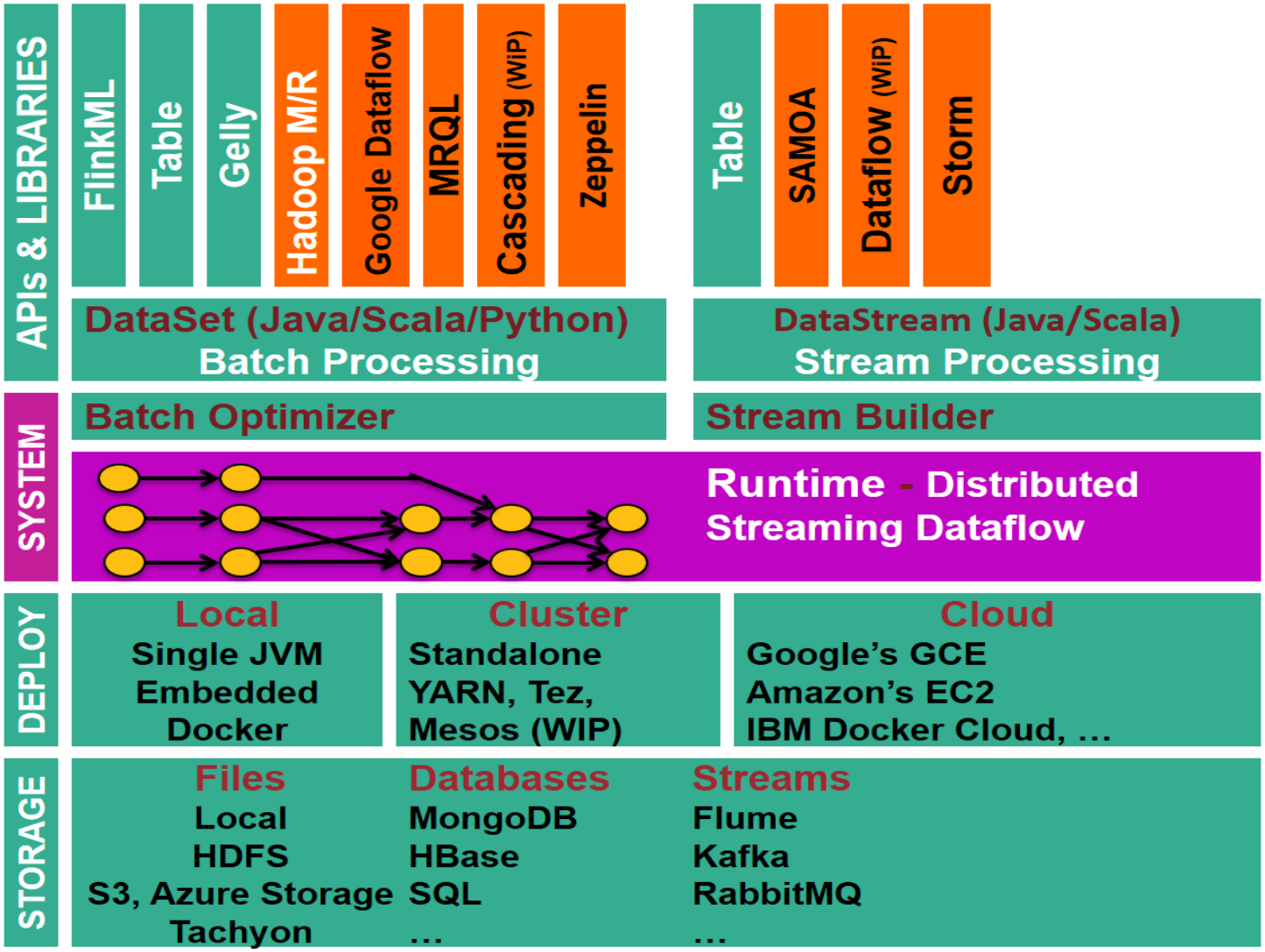
Apache Flink

- Apache Flink is an open source platform
  - Distributed stream and batch data processing."
    - https://flink.apache.org/
- The definition in wikipedia:
  - https://en.wikipedia.org/wiki/Apache_Flink

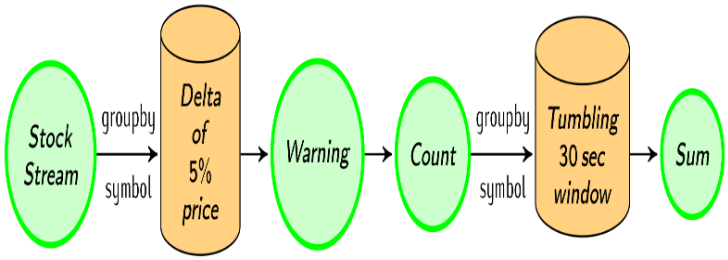
Apache Flink (Cont.)

- written in Java and Scala, consists of:
  - Big data processing engine:
  - Distributed and scalable streaming dataflow engine
- Several APIs in Java/Scala/Python:
  - DataSet API – Batch processing
  - DataStream API – Real-Time streaming analytics
- Domain-Specific Libraries:
  - FlinkML: Machine Learning Library for Flink
  - Gelly: Graph Library for Flink
  - Table: Relational Queries
  - FlinkCEP: Complex Event Processing for Flink

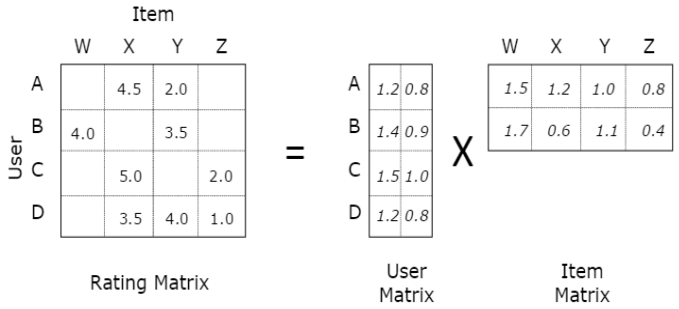
Apache Flink (Cont.)



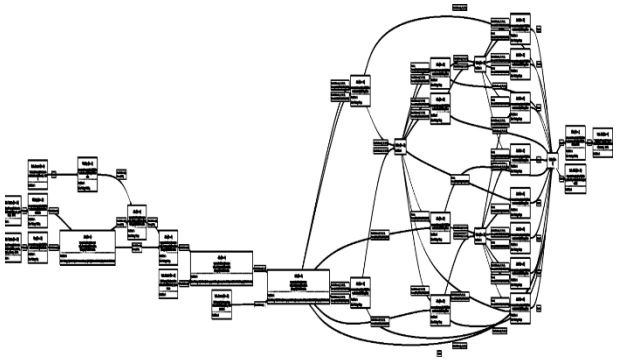
Apache Flink (Cont.)



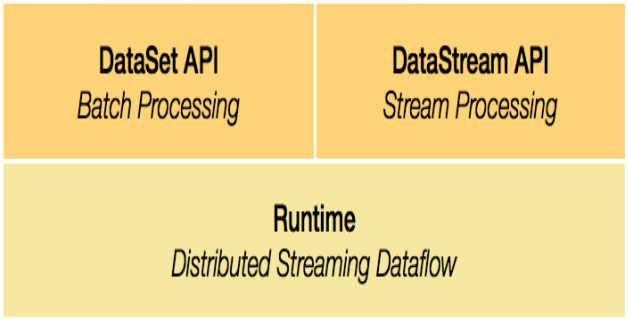
Real-Time stream processing

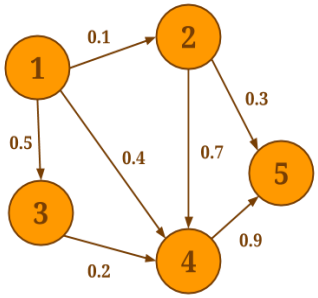


Machine Learning



Batch Processing





Graph Analysis

Big Data and Cloud

- **Cloud Computing Platform & Services**
  - (Cloudera, Hortonworks, MapR, Azure)

Big Data and NoSQL

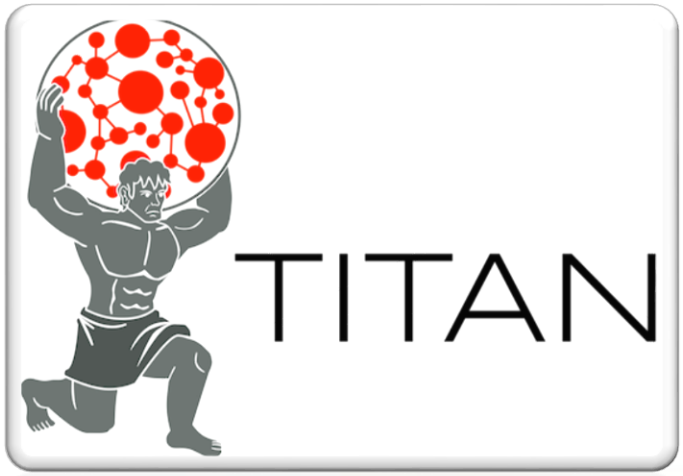
- **Key-values Stores**
  - **Unique key and a pointer to a particular item of data.**
    - Tokyo Cabinet/Tyrant, Redis,
    - Voldemort, *Oracle BDB (Oracle Big Data Solutions)* ,
    - Amazon SimpleDB, Riak
- **Column Family Stores**
  - **Very large amounts of data distributed over *Many* machines.**
    - Cassandra, HBase

Big Data and NoSQL (Cont.)

- Document Databases
  - Similar to key-value stores,
  - Semi-structured documents are stored in formats like JSON
  - Allowing nested values associated with each key.
  - Document databases support querying more efficiently.
    - CouchDB, MongoDb

Big Data and NoSQL (Cont.)

- Graph Database
- Flexible graph model
  - Instead of tables of rows and columns and the rigid structure of SQL
- Scale across multiple machines (Scale Out)
- Neo4J, InfoGrid, Infinite Graph, Titan



MongoDB Intro

- MongoDB Introduction
  - MongoDB is a document oriented
  - Leading NoSQL database

- MongoDB Characteristics
  - High availability and performance
  - Horizontally scalable
  - Flexible(dynamic) schemas
  - Deep query ability

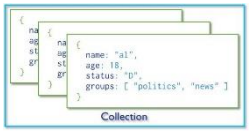
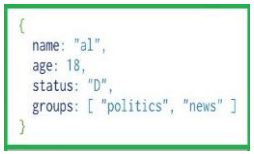
MongoDB Intro (Cont.)

- **No complex joins**
- **Tuning**
- Uses **internal memory** for storing the data
- Data is stored in **JSON** form
- **Replication** and **Sharding**
- **Indexing any attribute**
- Can be used with **JavaScript**

MongoDB Intro (Cont.)

- **Where to Use?**
  - Big Data
  - Content Management
  - Mobile and Social Infrastructure
  - User Data Management
  - Data Hub
- **Document and Collection concept:**
  - A document is a set of key-value pairs
  - Collection is a group of MongoDB documents
  - Documents within a collection with different fields

MongoDB Intro (Cont.)

SQL	MongoDB
database 	database 
table 	collection 
row 	document 
column	field
Table Join	Embedded Documents

MongoDB Intro (Cont.)

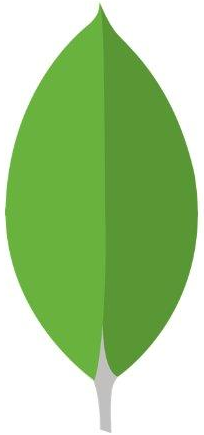
JSON Format

RDBMS		NoSQL
Databae		Database
Table, View		Collection
Row		Document (JSON, BSON)
Column		Field
Index		Index
Join		Embedded Document
Foreign Key		Reference
Partition		Shard

```
> db.user.findOne({age:39})
{
        "_id" : ObjectId("5114e0bd42…"),
        "first" : "John",
        "last" : "Doe",
        "age" : 39,
       "interests" : [
                "Reading",
                "Mountain Biking ]
       "favorites": {
                "color": "Blue",
                "sport": "Soccer"}
}
```

MongoDB- Datatypes

- **String**
  - This is the most commonly used datatype to store the data
  - String in MongoDB must be UTF-8 valid
- **Integer**
  - This type is used to store a numerical value
  - Integer can be 32 bit or 64 bit depending upon your server
- **Boolean**
  - This type is used to store a Boolean (true/ false) value
- **Double**
  - This type is used to store floating point values
- **Min/ Max keys**
  - This type is used to compare a value
    - Against the lowest and highest BSON elements



MongoDB– Datatypes (Cont.)

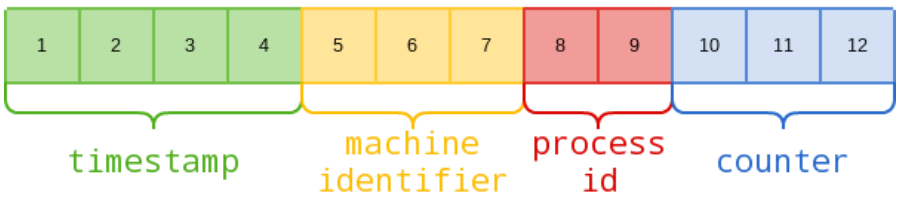
- **Arrays**
  - This type is used to store arrays or list or multiple values into one key
- **Timestamp**
  - Timestamp
    - Can be handy for recording when a document has been modified or added
- **Object**
  - This datatype is used for embedded documents
- **Null**
  - This type is used to store a Null value
- **Symbol**
  - This datatype is used identically to a string
  - It's generally reserved for languages that use a specific symbol type
- **Object ID**
  - This datatype is used to store the document's ID

MongoDB– Datatypes (Cont.)

- **Date**
  - Used to store the current date or time in UNIX time format
  - You can specify your own date time
    - By creating object of Date and passing day, month, year into it
- **Binary data**
  - This datatype is used to store binary data
- **Code**
  - This datatype is used to store JavaScript code into the document
- **Regular expression**
  - This datatype is used to store regular expression

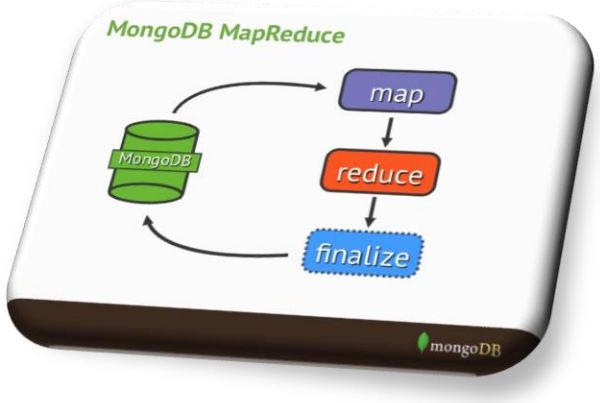
MongoDB– ObjectID

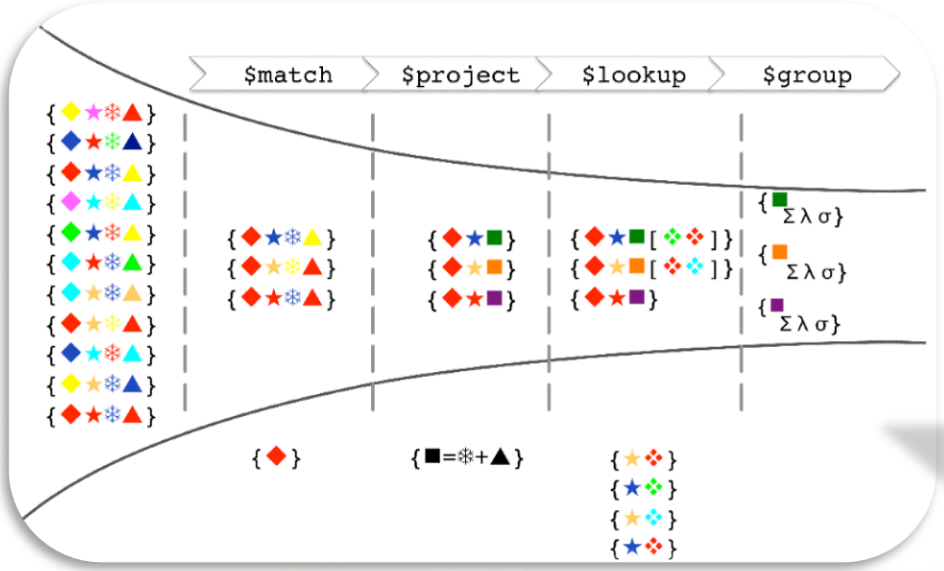
- **Object Id**
  - _id is 12 bytes hexadecimal number
  - Unique for every document in a collection
  - 12 bytes are divided as follows
    - 4 bytes timestamp (sec)
    - 3 bytes machine id
    - 2 bytes process id
    - 3 bytes incrementer
      - Incrementing value starting with a random number
  - If we don't specify the _id parameter
    - MongoDB assigns a unique ObjectId for this document

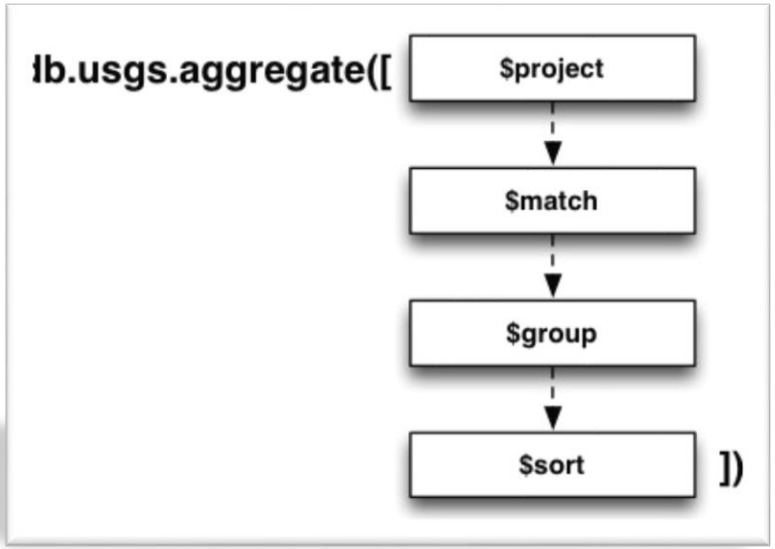


MongoDB– Aggregation

- Aggregation Pipeline
- Map-Reduce
- Single Purpose Aggregation Operations





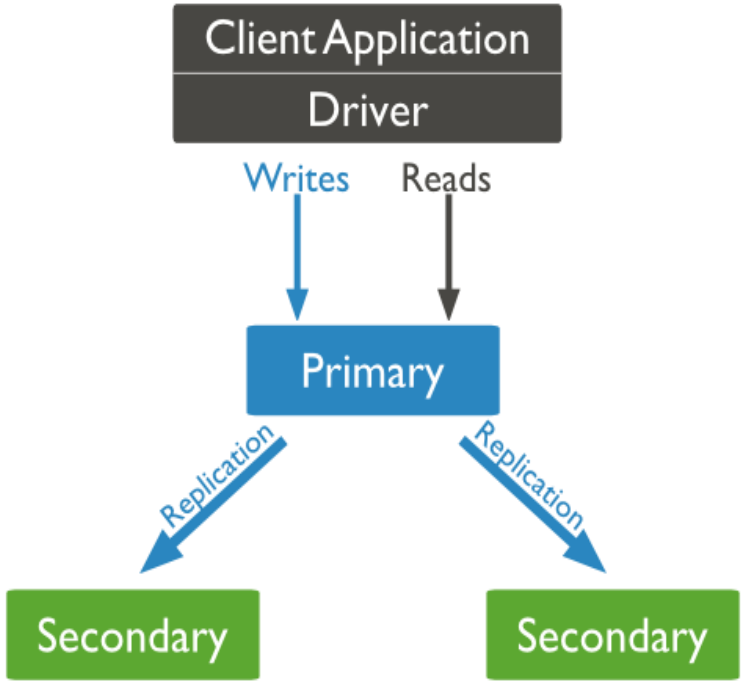


MongoDB- Replication

- **How Replication Works in MongoDB**

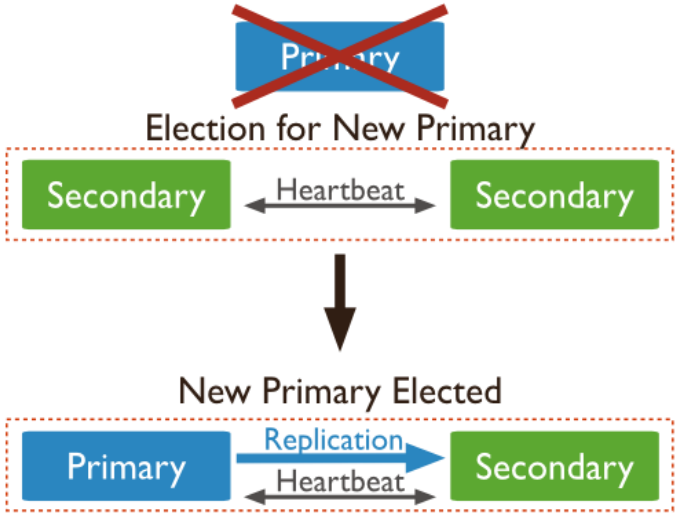
- MongoDB achieves replication by the use of **replica set**
  - Replica set is a group of two or more nodes
    - Generally minimum 3 nodes are required

- **In a replica set**
  - One node is primary node
    - Receives all write operations
  - Remaining nodes are secondary
    - Apply operations from the primary
    - They have the same data set



MongoDB- Replication (Cont.)

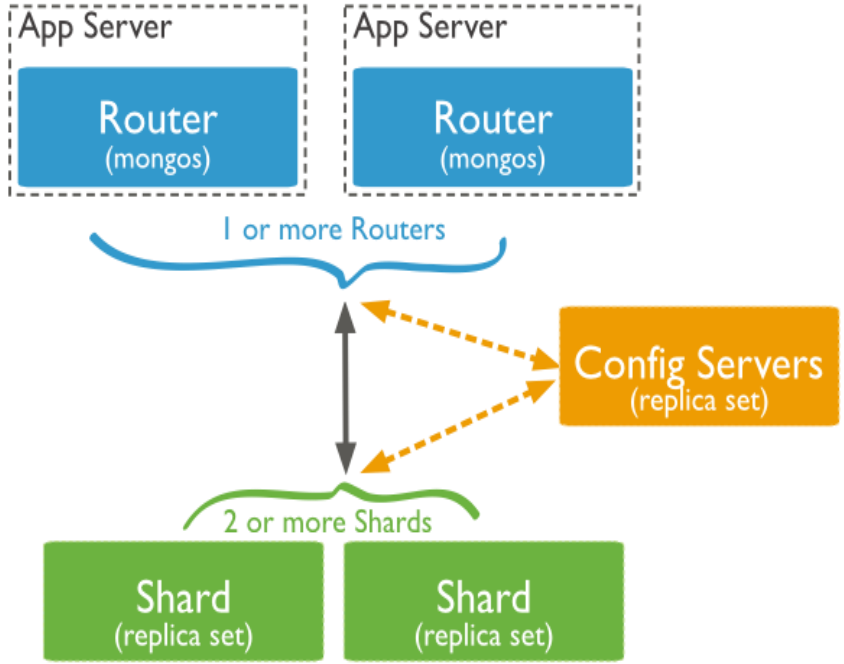
- All data replicates from primary to secondary node
- At the time of automatic failover or maintenance
  - Election establishes for primary and a new primary node is elected
- After the recovery of failed node
  - It again join the replica set and works as a secondary node



MongoDB- Sharding

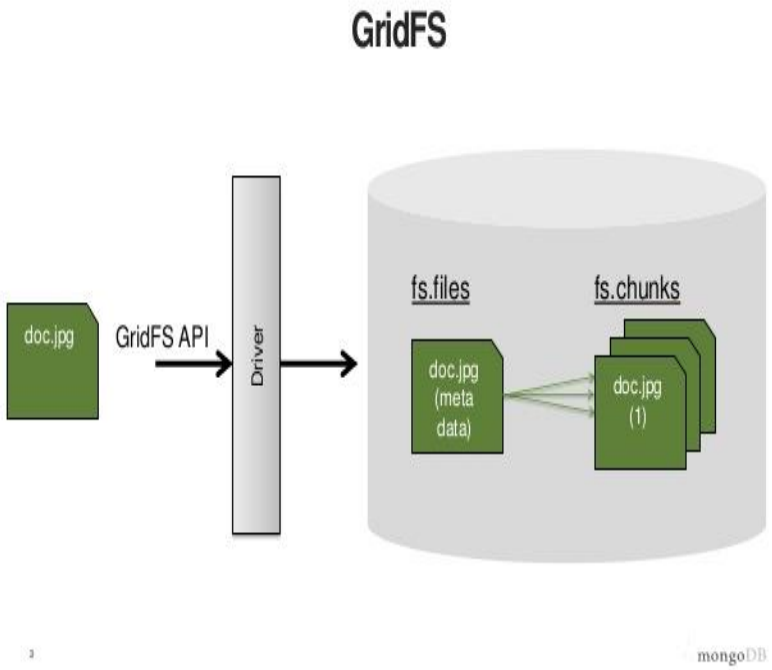
- Distributing data and its replications across multiple machines

- Why Sharding:
  - Handling large data sets
  - High throughput operations
  - Horizontal scaling



MongoDB -GridFS

- MongoDB specification for storing and retrieving large files
  - Such as images, audio files, video files, etc.
- It is kind of a file system to store files
- Store files even greater than its document size
  - Limit of 16MB



MongoDB Atlas

MongoDB Atlas provides all of the features of MongoDB without the operational heavy lifting required for any application.

Features:

- ✓ MongoDB Atlas is a **cloud service**
- ✓ For:
  - Running
  - Monitoring
  - Maintaining

  MongoDB deployments

- ✓ Allows to quickly provision it in the cloud and only pay an hourly fee
- ✓ Including the provisioning of dedicated servers for the MongoDB instances.

MongoDB Atlas (cont.)

Atlas provides:

- ✓ Security features to protect access to your data
- ✓ Built in replication for always-on availability
- ✓ Tolerating complete data center failure
- ✓ Backups and point in time recovery to protect against data corruption
- ✓ Fine-gained monitoring to let you know when to scale
- ✓ A scale choice of providers, regions and billing options

MongoDB Atlas (cont.)

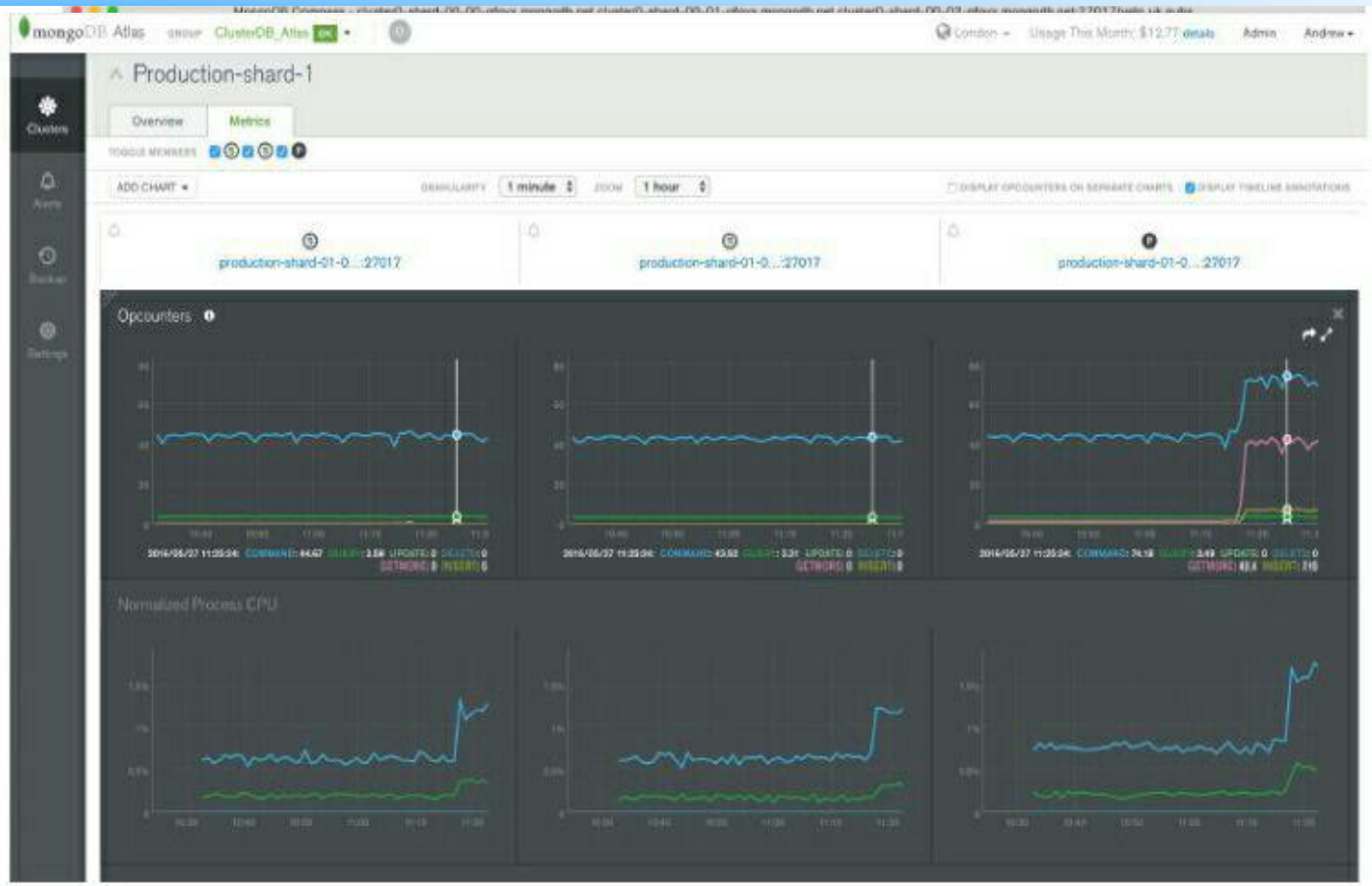


Figure 5: Database monitoring with MongoDB Atlas GUI

MongoDB Atlas (cont.)

Link:

https://www.mongodb.com/cloud/atlas/faq

MongoDB Compass

- ✓ Easily analyze without requiring knowledge of MongoDB query syntax

Features:

- MongoDB Compass provides users with a graphical view of their MongoDB
- It minimizes performance impact on the database and can produce results quickly
- Write queries to reverse engineer the document structure, field name and data types

MongoDB Compass (cont.)

Features:

- Insert/edit/delete/clone documents through the GUI
- Build and visually interact with geo/coordinate data to construct queries in a few clicks of a button
- Visual explain plans to understand the performance of a query
- SSH tunnels to allow users to connect securely from outside of a datacenter firewall
- Can be use for free during development

MongoDB Compass(cont.)

- Can be use for free during development

- It is available for production use with MongoDB professional or MongoDB Enterprise Advanced subscriptions

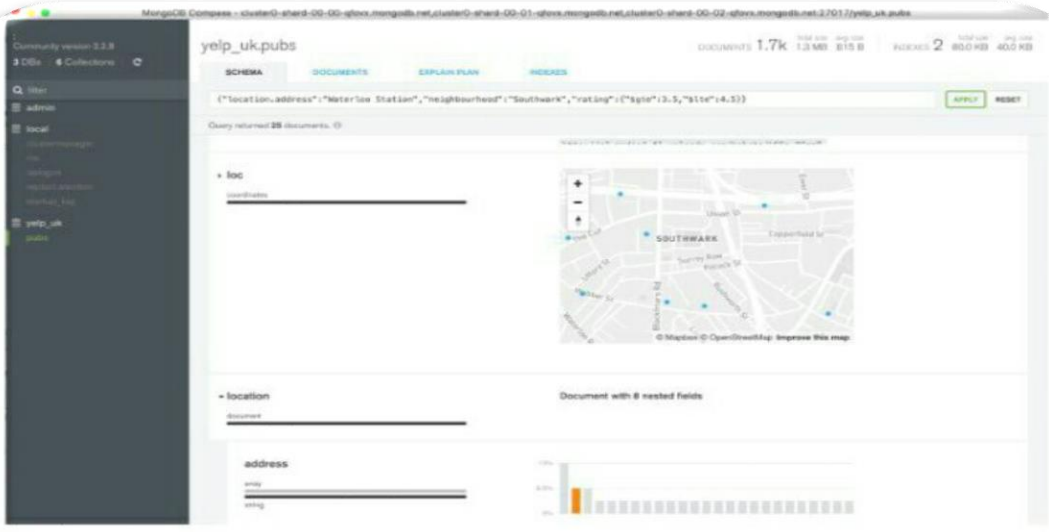


Figure 1: View schema & interactively build and execute database queries with MongoDB Compass

MongoDB Compass(cont.)

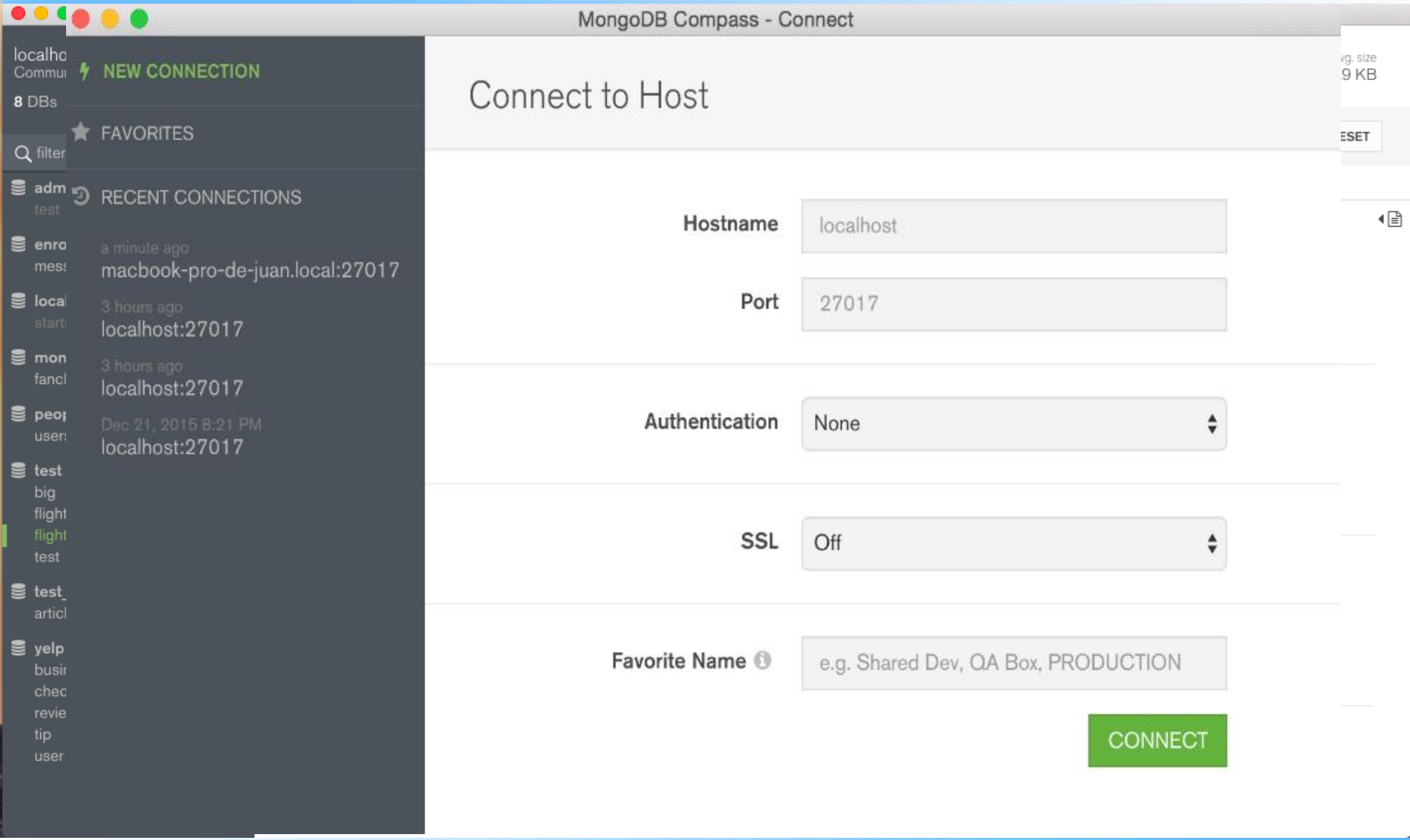
Benefit:

- Faster time to market
- Easier project handoffs
- Increased productivity

- **MongoDB Compass for windows:**

https://www.mongodb.com/download-center?jmp=nav#compass

MongoDB Compass(cont.)



MongoDB driver

Compatible with:

- C 
- C++ 
- C# 
- Java 
- Node.js 
- Ruby 
- Scala 
- Php 
- **Python** 
- Perl 

Full-text search

- MongoDB began with supporting **Full-Text Search** using **Text Indexes**
- It has now become an **integral part of the product**
- It refers to the **full-text database** against the search criteria specified by the user
- It is not proposed as a complete replacement of search engine databases
- It used for applications that are built with MongoDB

Index Type

Features:

- Indexes support the efficient resolution of queries
- Without indexes, MongoDB must scan every document of a collection
- Indexes are special data structure
- Stores a small portion of data set in any easy to traverse form
- Stores the value of specific field

Index Type

- **MongoDB provides different types of indexes for different purposes and different types of content.**

Types:

- **Single Field Indexes:** only includes data from a single field of the documents
- **Compound Indexes:** includes more than one field of the documents
- **Multikey Indexes:** is an index on an array field, adding an index key for each value

mongoDB. Index Type (cont.)

- **Geospatial Indexes and Queries:**
  - Support location-based searches on data
  - Data store as either GeoJSON objects
- **Text Indexes:** Text indexes support search of string content in documents
- **Hashed Index:**
  - Maintain entries with hashes of the values of the indexed field
  - Are primarily used with sharded clusters to support hashed shard keys
- **Index Properties:** The properties you can specify when building indexes
- **TTL Indexes:** used for TTL collections, which expire data after a period of time

mongoDB. Index Type (cont.)

- **Unique Indexes:** Causes to reject all documents that contain a duplicate value for the indexed field
- **Sparse Indexes:** Does not index documents that do not have the indexed field
- **Index Creation:** The options available when creating indexes
- **Index Intersection:** The use of index intersection to fulfill a query
- **Multikey Index Bounds:** The computation of bounds on a multikey index scan

Companies that use MongoDB

- Twitter
- Castlight Health
- Thumbtack
- IBM
- Citrix
- T-Mobile
- Zendesk
- Sony
- HTC
- Techstars
- Atlassian
- Udacity
- BrightRoll
- RetailMeNot
- Hootsuite
- SurveyMonkey
- Shyp
- Criteo
- MuleSoft
- HackerRank

Few clients

- For search suggestions
- Metadata storage
- Cloud management
- Merchandizing categorization

- To archive billions of records

- For back-end storage SourceForge front pages,
- Project pages,
- And download pages for all projects

- To store venues and user ‘check-inns’ into venues,
- Sharding the data over more than 25 machines on Amazon EC2

Pros and Cons - MongoDB

Advantages

- Performance
- Document Model
- Flexible Schema

mongoDB

Disadvantages

- No transaction
- No join
- Memory limitation

Pros and Cons – MongoDB (Cont.)

- **The philosophy behind MongoDB :**

✓ To retain as many functionalities as possible

✓ While permitting horizontal scale

✓ Make developer's life easier.

Pros and Cons – MongoDB (Cont.)

Advantages :

AGPL license

- ✓ Every changes to the Mongodb is open sourced to the community.
- ✓ Designed for big data storage and query
- ✓ Mongodb is a best fit in presence of a lot of data

Pros and Cons – MongoDB (Cont.)

Aims at Social Network applications, Text Mining, Search Engine, Ad hoc Documents.

Easy to scale out

- ✓ Its a breeze in NoSQL database like MongoDB.

  Scale horizontally is hard task in relational database
- ✓ Add as the data and traffic grows
- ✓ Keep the responsive speed and availability.

Pros and Cons – MongoDB (Cont.)

Document as basic storage unit

- ✓ A document is just a simple JSON like object,
- ✓ BSON in Mongodb,
- ✓ Indexing which implemented by B-Tree,

  Index ; on unique field or multiple fields,

Pros and Cons – MongoDB (Cont.)

No schema in MongoDB

- ✓ Document can have any number of fields
- ✓ Fit into the OOP model

  Most programming language support today.
- ✓ Do a quick developing or prototyping (Python, Ruby and PHP)

Pros and Cons – MongoDB (Cont.)

Optional strict consistent level

- ✓ Insertions operation is fire-and-forget
- ✓ MongoDB supports auto sharding and auto failover,

Simple Querying

- ✓ Everything from that one document,
- ✓ No reference to other documents
- ✓ No concept of tables, rows, SQL, schemas

Pros and Cons – MongoDB (Cont.)

- **Disadvantages :**

Not support transaction

- ✓ They either all done, or nothing is done.
- ✓ RDBMS do this by transaction.
- ✓ The relation between MongoDB documents is weak.
- ✓ Stream of dependent events (Bank Account !!)
- ✓ Many related bank data needs to be modified at the same time.

Pros and Cons – MongoDB (Cont.)

Not support join operation

RAM limitation

- ✓ Size of your database is Limited by virtual memory provided by Operating System and hardware.
- ✓ For production environment, a 64bits system is a must.

What is MongoDB great for?

- ✓ RDBMS replacement for Web Applications.
- ✓ Semi-structured Content Management.
- ✓ Real-time Analytics & High-Speed Logging.
- ✓ Caching and High-Scalability

Web 2.0, Media, SAAS, Gaming

HealthCare, Finance, Telecom, Government

mongoDB. Not great for?

- Highly Transactional Application
- Problems requiring SQL

MongoDB- Security

- *Non-relational data stores*

  Think NoSQL databases, which by themselves usually lack security

  (which is instead provided, sort of, via middleware).

MongoDB- Security (Cont.)

NoSQL(Mongo DB) Security

- History
- Authentication
- Authorization
- Auditing
- Transport Encryption – SSL
- MongoDB Secure Development Lifecycle
- Documentation and Notifications
- Future Work

MongoDB- Security (Cont.)

History and Three A"s

- 2.4 offers a much better story around security
- Investing very heavily right now.

- Authentication

  Who are you?

- Authorization

  What can you do?

- Auditing

  What have you done?

Authentication

Authentication is about proving “who” you are.

❑ Password Authentication:

- This is the only authentication mechanism available in MongoDB version 2.2 and prior
- Still the only version available in the free product
- In 2.4+ this mechanism is called MONGODB-CR

Password Authentication

Use one-way function F

I am “username”, let me in

Prove it, here is a random # N

Here is F(N, hash(<mypwd>))

Nobody else could know that, welcome back!

mongod

Knows only my password hash

Hash never transmitted over the network!

External Authentication

- **Use common / standardized authentication**
- **SASL: Simple Authentication and Security Layer**
  - Framework for building authentication
  - MongoDB uses the Cyrus sasl2 library
- **Kerberos (available in the Enterprise Edition)**
  - GSSAPI
    - Driver support in python, java, C#, Node.js, perl

MongoDB- Security (Cont.)

Granting privileges

```
# mongo mongodb.mycompany.com
> use appDB;
> db.system.users.find();
    {
        "_id": ObjectId("519e842804f5f7f7921dbf89"),
        "user": "spencer"
        "userSource": "$external",
        "roles": ["readWrite",  "dbAdmin”]
    }
```

MongoDB- Security (Cont.)

Authorization

Once MongoDB has established “who” you are, authorization is about determining “what” you are allowed to do.

MongoDB- Security (Cont.)

Authorization Roles in 2.2 and Prior

- Database level read-only
- Database level read-write
- System-wide read-only
- System-wide read-write

Sample user document:

```
> db.system.users.find().pretty()
{
        "_id": ObjectId("519e842804f5f7f7921dbf89"),
        "user": "spencer"
        "pwd": "22c83553ed7ce252d8b0c9f716cae4de",
        "readOnly": false
}
```

Authorization Roles in 2.4

- read
- readWrite
- dbAdmin
- userAdmin
- **readAnyDatabase**
- **readWriteAnyDatabase**
- **dbAdminAnyDatabase**
- **userAdminAnyDatabase**
- **clusterAdmin**

The roles that are **bold** can only be granted in the **admin** database.

userAdmin

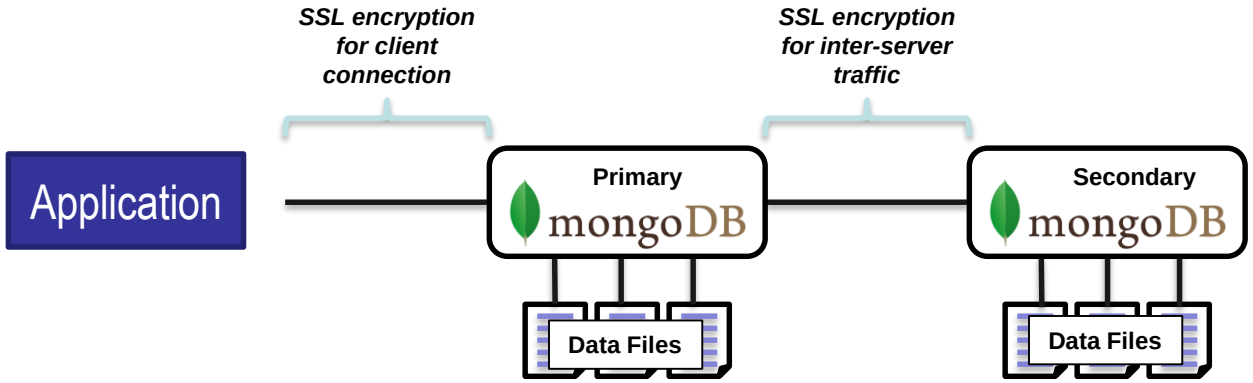
- The userAdmin role on database “foo” lets you grant **any** db-level role to **any** user from the “foo” database (including yourself).
- The userAdminAnyDatabase role lets you grant **any** role in the system to **any** user (including yourself).
- This means they can be used to grant yourself roles you didn’t previously have!
- This makes userAdmin **effectively** a super-user
- Access to these roles should be carefully controlled!

Auditing

Monitor user activity:

- userID added to standard output in 2.4
- No separate audit log
- Much more coming in 2.6

Transport Encryption - SSL



http://docs.mongodb.org/manual/administration/ssl/

Future

- User-defined roles
- Collection level access control
- Field level access control
- Auditing
- X.509 authentication, for both user and intra-cluster authentication.
- External configuration of user's roles (LDAP)

MongoDB - Resources

- www.docs.mongodb.com
- www.tutorialspoint.com/mongodb
- www.codeschool.com/courses/the-magical-marvels-of-mongodb

The time goes on…

Started in 2004

Released in 2010

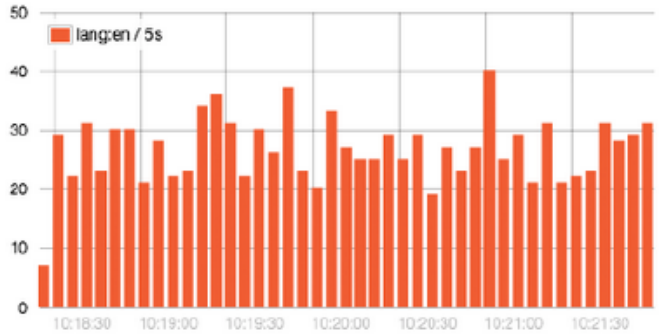
In 2014, 70$ million in Series C funding

And now here is the 5.5.0 release with great components!

What it gives to us?(cont.)

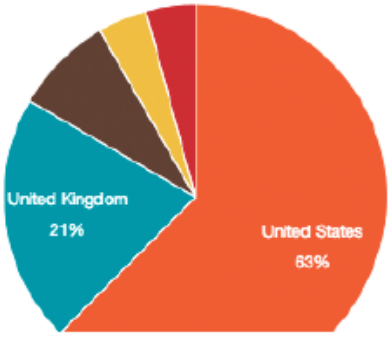
real time data

Data flows into your system all the time. The question is ... how quickly can that data become an insight? With Elasticsearch, real-time is the only time.



real time analytics

Search isn't just free text search anymore - it's about exploring your data. Understanding it. Gaining insights that will make your business better or improve your product.

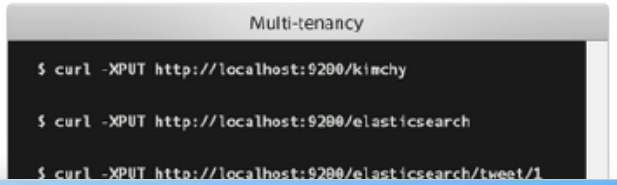


high availability

Elasticsearch clusters are resilient - they will detect and remove failed nodes, and reorganise themselves to ensure that your data is safe and accessible.

multi-tenancy

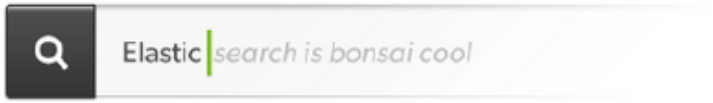
A cluster can host multiple indices which can be queried independently or as a group. Index aliases allow you to add indexes on the fly, while being transparent to your application.



What it gives to us?(cont.)

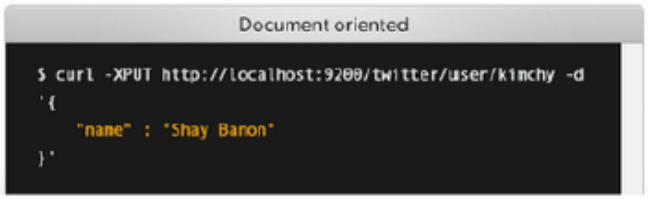
full text search

Elasticsearch uses Lucene under the covers to provide the most powerful full text search capabilities available in any open source product. Search comes with multi-language support, a powerful query language, support for geolocation, context aware did-you-mean suggestions, autocomplete and search snippets.



document oriented

Store complex real world entities in Elasticsearch as structured JSON documents. All fields are indexed by default, and all the indices can be used in a single query, to return results at breath taking speed.



conflict management

Optimistic version control can be used where needed to ensure that data is never lost due to conflicting changes from multiple processes

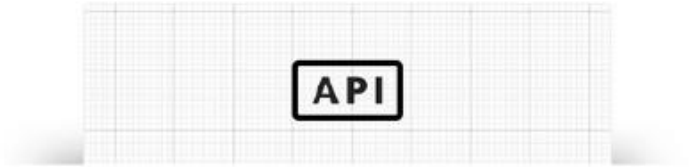
schema free

Elasticsearch allows you to get started easily. Toss it a JSON document and it will try to detect the data structure, index the data and make it searchable. Later, apply your domain specific knowledge of your data to customise how your data is indexed.

What it gives to us?

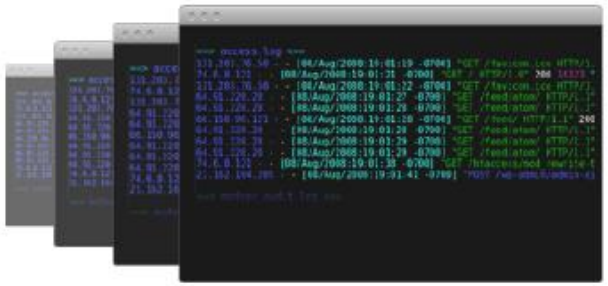
restful api

Elasticsearch is API driven. Almost any action can be performed using a simple RESTful API using JSON over HTTP. An API already exists in the language of your choice.



per-operation persistence

Elasticsearch puts your data safety first. Document changes are recorded in transaction logs on multiple nodes in the cluster to minimise the chance of any data loss.



apache 2 open source license

Elasticsearch can be downloaded, used and modified free of charge. It is available under the Apache 2 license, one of the most flexible open source licenses available.

build on top of apache lucene™

Apache Lucene is a high performance, full-featured Information Retrieval library, written in Java. Elasticsearch uses Lucene internally to build its state of the art distributed search and analytics capabilities.

Who else uses it?(cont.)

Supporting e-commerce search for 60+ countries in 21+languages

Providing search on azure and powering social dynamics

Searching Across 800 million listings in sub seconds

Generate Actionable value from game play data and server events

Delivering a better help experience for over a billion users

Jet Propulsion Laboratory
California Institute of Technology

Powering the search for Interplanetary discovery

Who else uses it?

slack

Supporting e-commerce search for 60+ countries in 21+languages

BBC

Unlocking yesterday's content for the future of media search

CISCO

Reducing system downtime at the basis of cisco's cloud native

NVIDIA

Enhancing user experience by processing over a billion of events every day

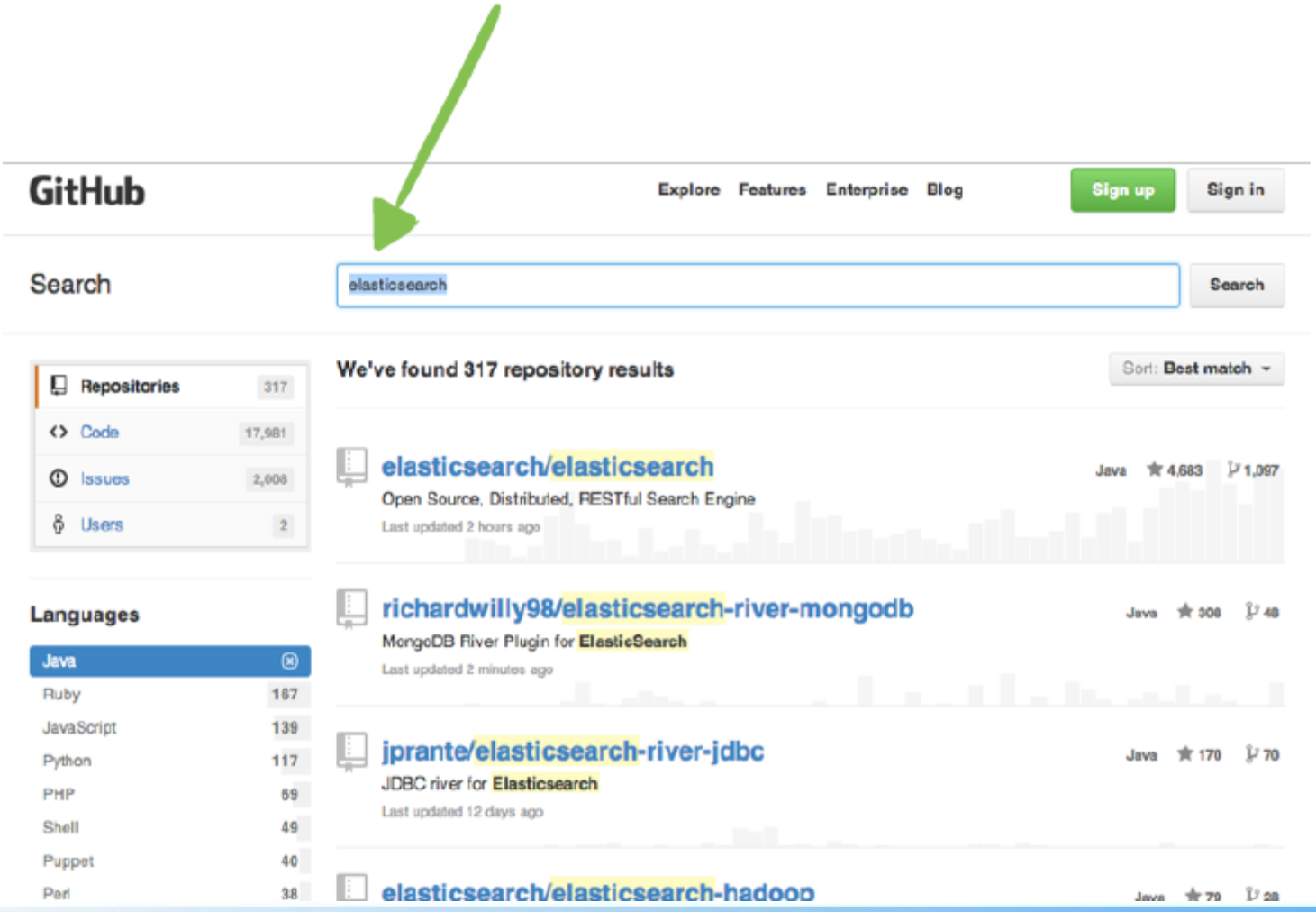
UBER

Aggregating business metrics to control critical marketplace behaviors

And many others…

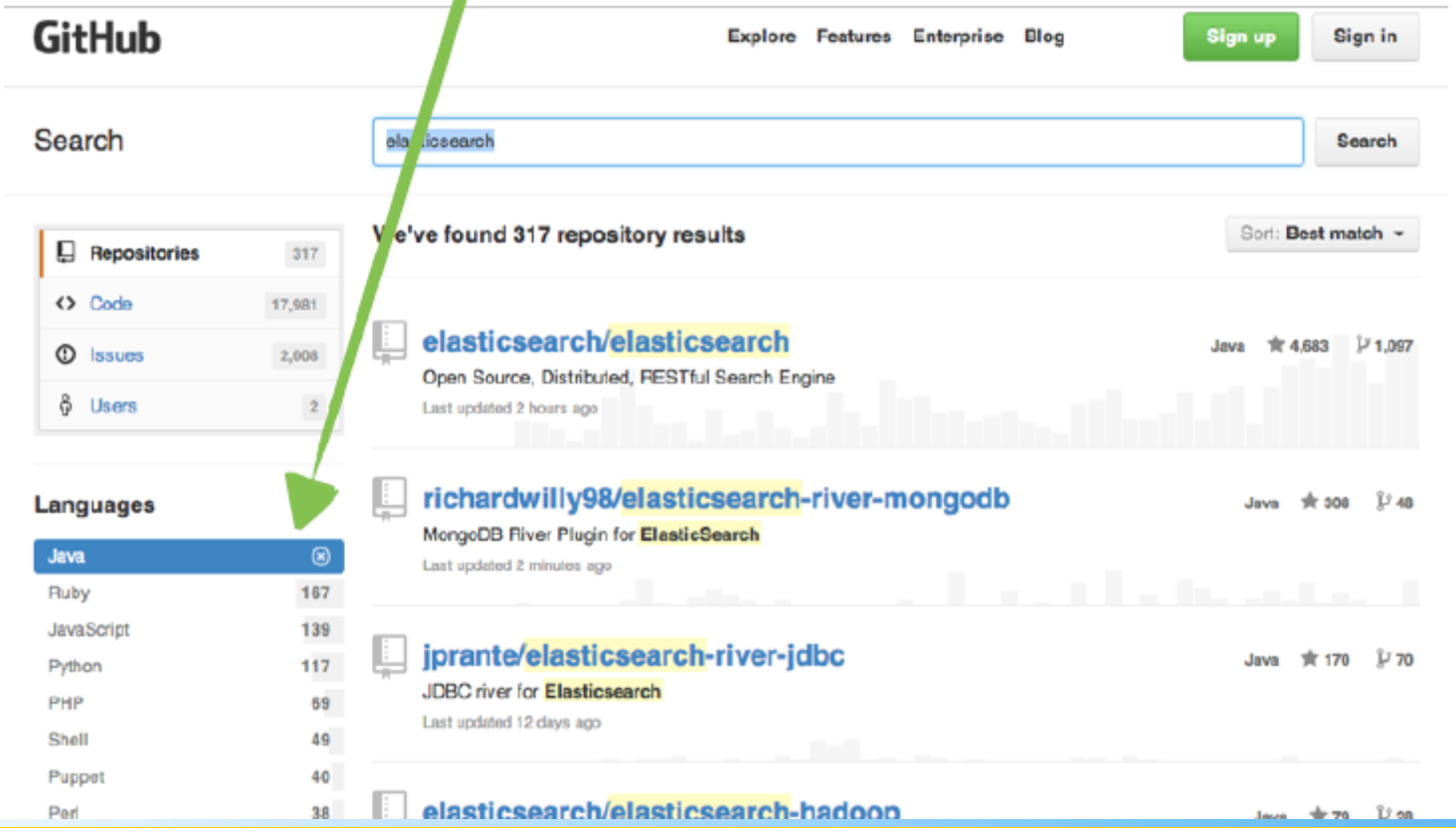
A case of usage…(cont.)

Unstructured search



A case of usage…(cont.)

Structured search



A case of usage…(cont.)

Enrichment

GitHub Explore Features Enterprise Blog Sign up Sign in

Search elasticsearch Search

Repositories 317
Code 17,981
Issues 2,008
Users 2

We've found 317 repository results

Sort: Best match

elasticsearch/elasticsearch Java ★ 4,683 1,097
Open Source, Distributed, RESTful Search Engine
Last updated 2 hours ago

richardwilly98/elasticsearch-river-mongodb Java ★ 308 48
MongoDB River Plugin for ElasticSearch
Last updated 2 minutes ago

jprante/elasticsearch-river-jdbc Java ★ 170 70
JDBC river for Elasticsearch
Last updated 12 days ago

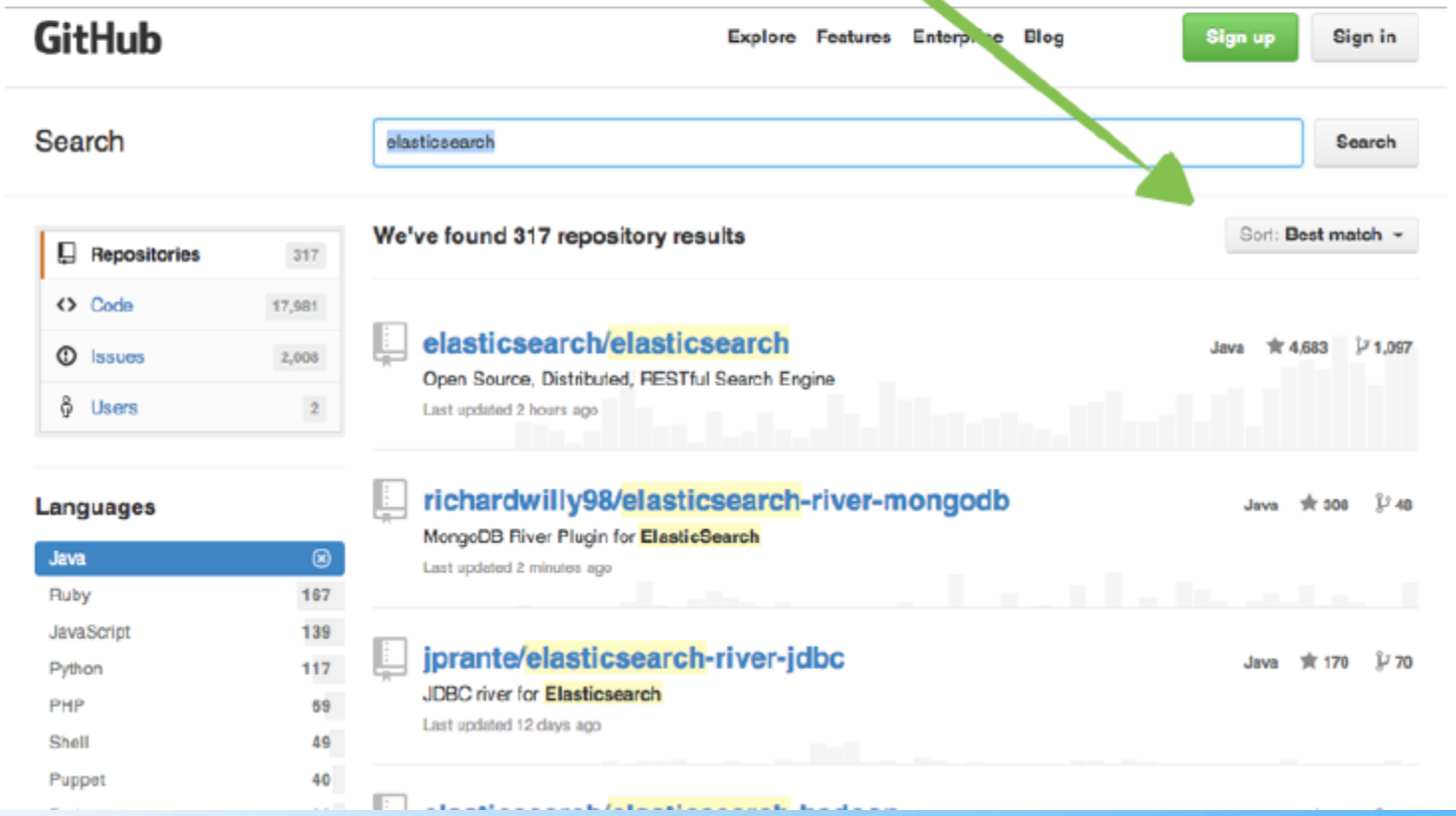
elasticsearch/elasticsearch-hadoop

Languages

Java
Ruby 167
JavaScript 139
Python 117
PHP 59
Shell 49
Puppet 40
Perl 38

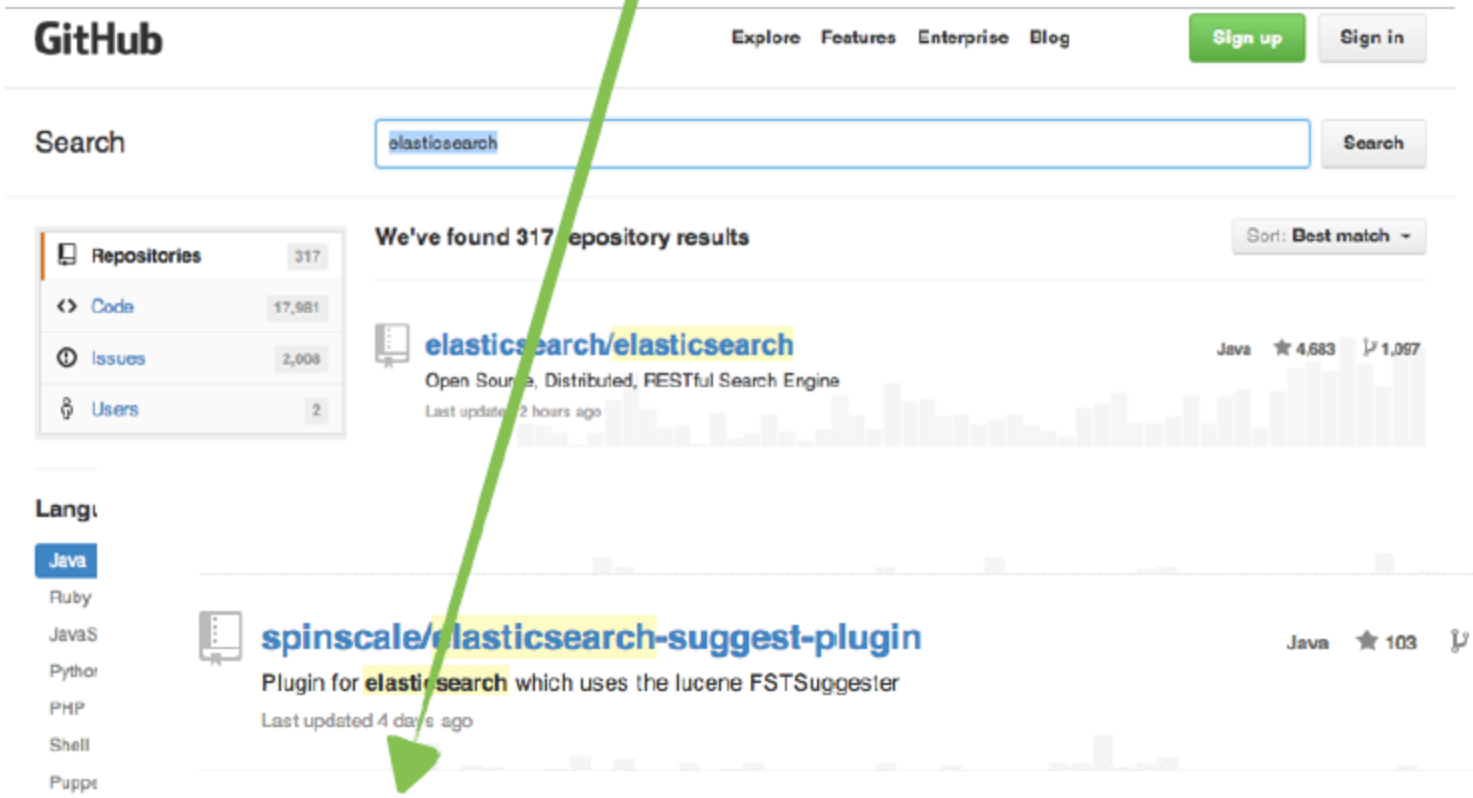
A case of usage…(cont.)

Sorting



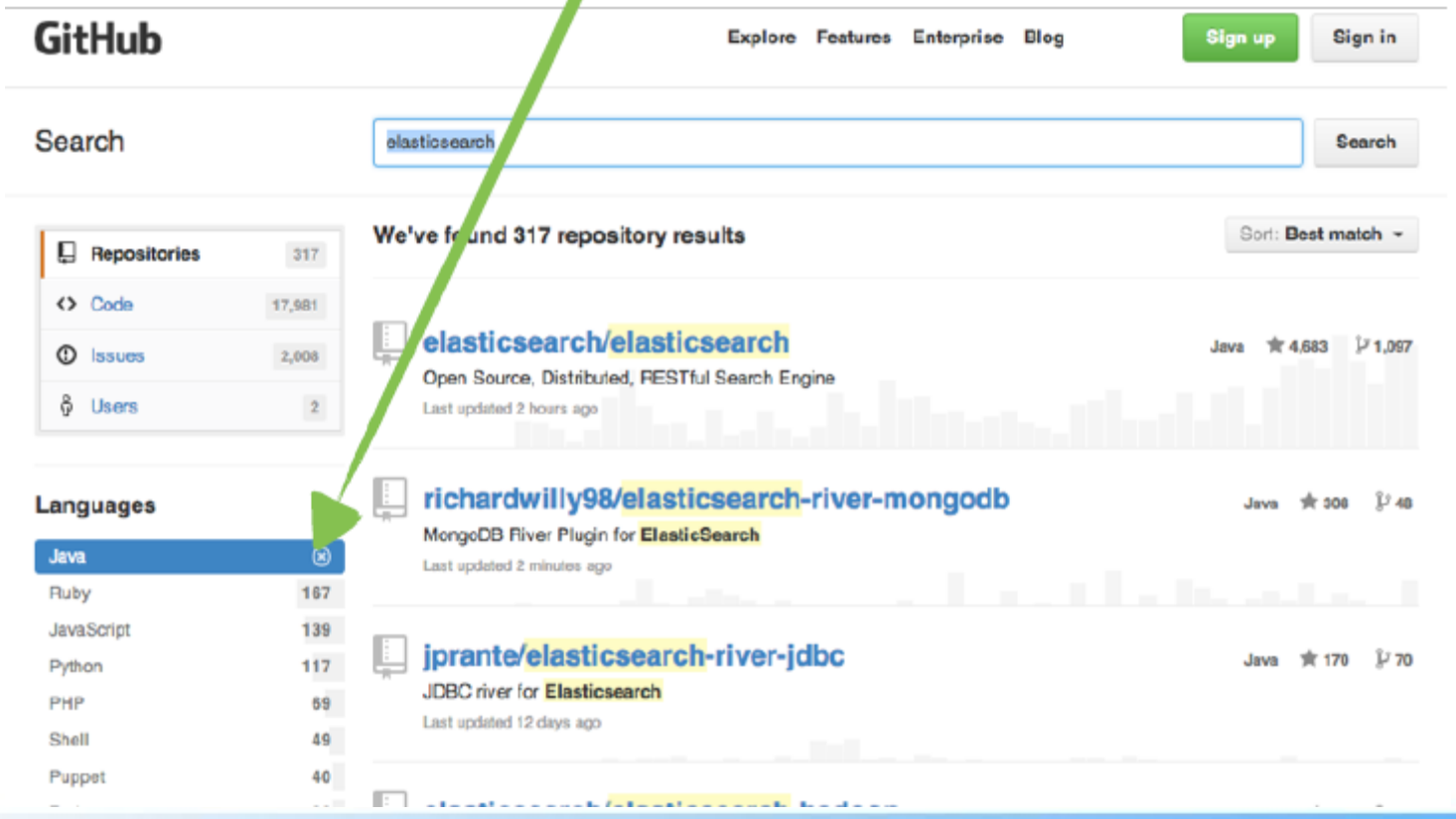
A case of usage…(cont.)

Pagination



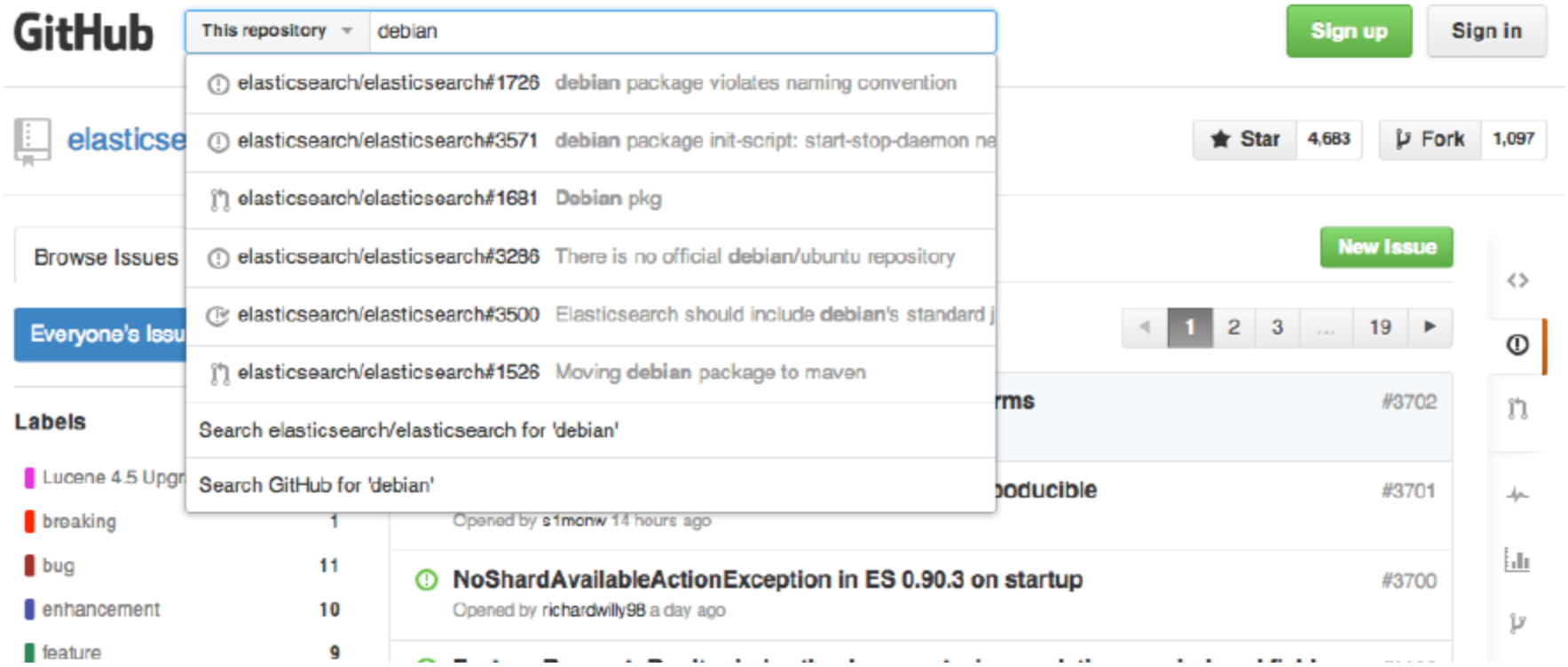
A case of usage…(cont.)

Aggregation



A case of usage…

Suggestions




ARTIFICIAL INTELLIGENCE


logstash

Capabilities

- Managing events and logs
- Collect data
- Parse data
- Enrich data
- Store data (for search and visualize)

Working structure

Managing events and logs

Collect data

} Input

Parse data

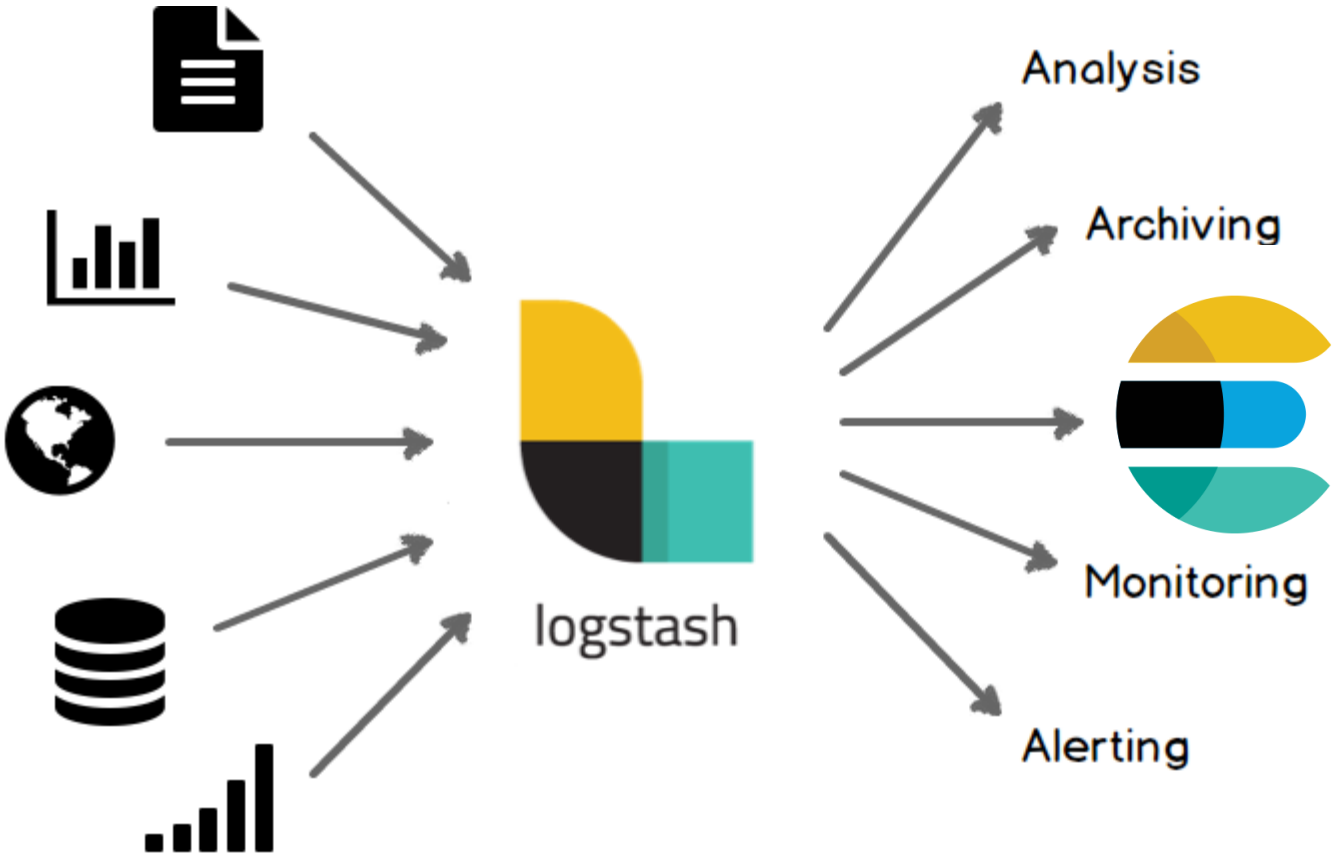
Enrich data

} Filter

Store data (search and visualise)

} Output

Architectural view

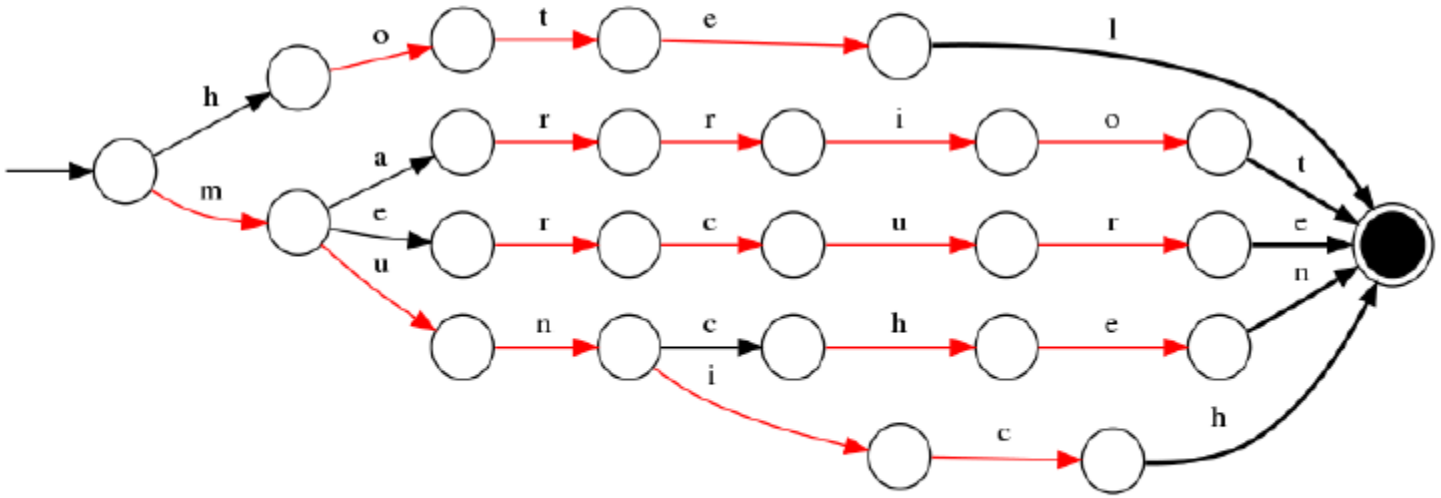


elasticsearch

Capabilities

- Store schema less data (create a schema for your data)
- Manipulate your data record by record (use multi-document APIs to o bulk ops)
- Perform Queries/Filters on your data for insights
- Use APIs to monitor the deployment site
- Built-in Full-Text-Search and analysis
- Auto completion
- Percolate API
- Scalability

Auto Completion: FST



Auto Completion

```
curl -X PUT localhost:9200/hotels/hotel/2 -d '
{
   "name" : "Hotel Monaco",
   "city" : "Munich",
   "name_suggest" : {
      "input" : [
         "Monaco Munich",
         "Hotel Monaco"
      ],
      "output": "Hotel Monaco",
      "weight": 10
   }
}'
```

Percolate API

Store the queries in Elastic search

Pass docs as queries

Observe matched queries

Percolate API

As a simple use case:

Assume that you tell the customer that he will be notified when plane ticket will be available and cheaper.

For this case you should store customer's criteria about desired flight.

When you store flight data, match it against saved percolators.

Percolate API

Store Query

```
curl -XPUT 'localhost:9200/my-index/.percolator/1' -d '{
    "query" : {
        "match" : {
            "message" : "bonsai tree"
        }
    }
}'
```

Match document

```
curl -XGET 'localhost:9200/my-index/my-type/_percolate'
-d '{
    "doc" : {
        "message" : "A new bonsai tree in the office"
    }
}'
```

Percolate API

```
{
    "took" : 19,
    "_shards" : {
        "total" : 5,
        "successful" : 5,
        "failed" : 0
    },
    "total" : 1,
    "matches" : [
        {
          "_index" : "my-index",
          "_id" : "1"
        }
    ]
}
```

Snapshot/Restore

Snapshot

```
curl -XPUT "localhost:9200/_snapshot/my_backup/snapshot_1?wait_for_completion=true"
```

Restore

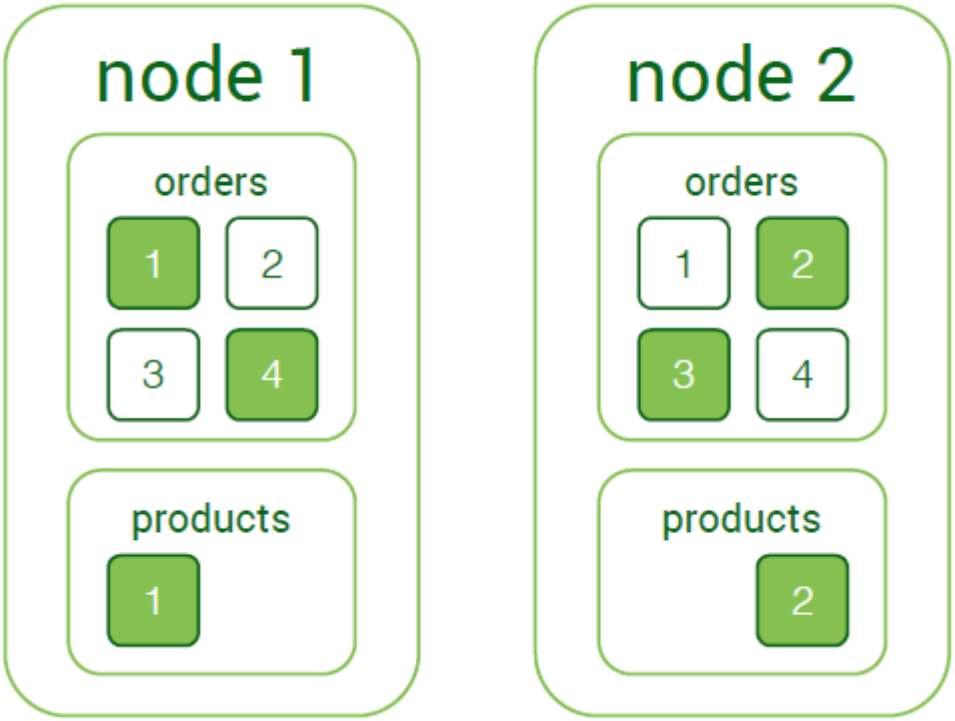
```
curl -XPOST "localhost:9200/_snapshot/my_backup/snapshot_1/_restore"
```

Distributed and scalable

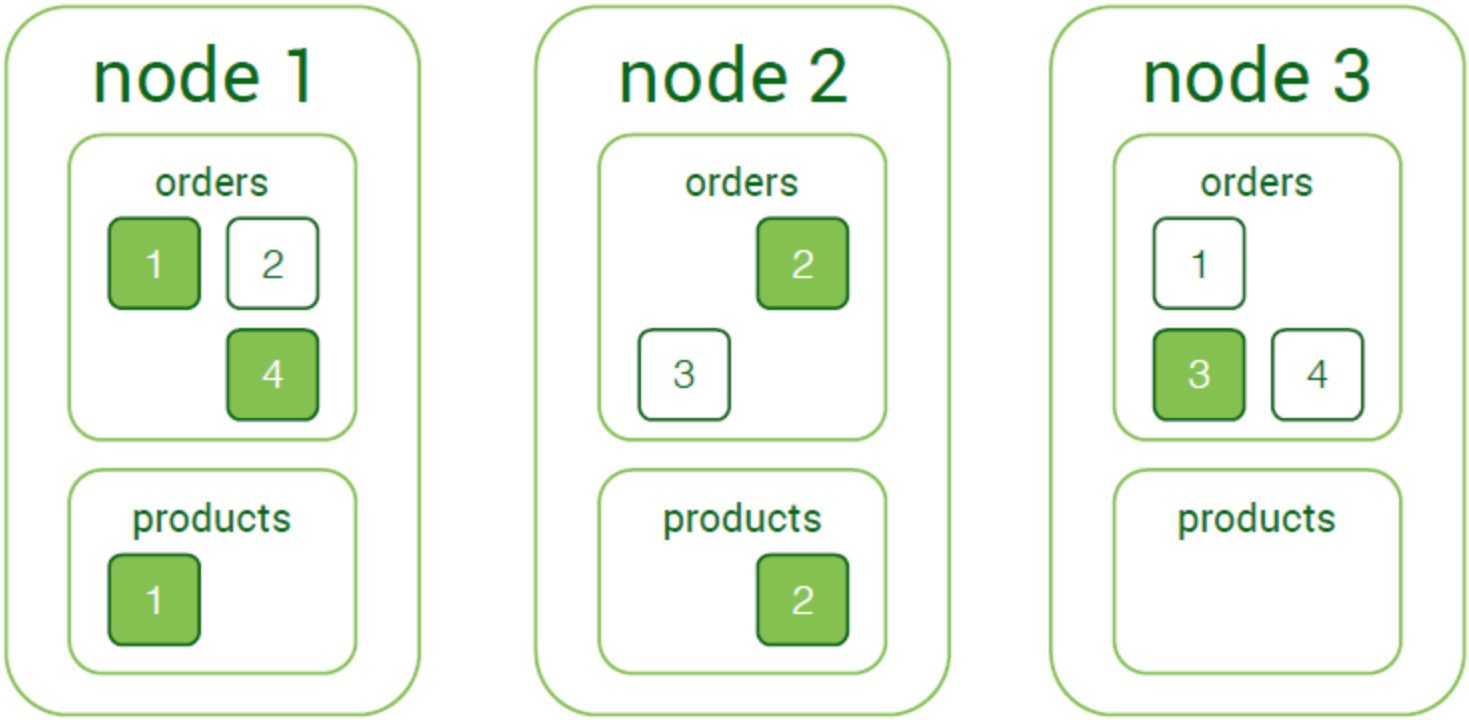

```
curl -X PUT localhost:9200/orders -d '{
    "settings.index.number_of_shards" :  4
    "settings.index.number_of_replicas": 1
}'
```

```
curl -X PUT localhost:9200/products -d '{
    "settings.index.number_of_shards" :  2
    "settings.index.number_of_replicas": 0
}'
```

Distributed and scalable



Distributed and scalable



API tour

Create

```
» curl -X PUT localhost:9200/books/book/1 -d '
{
    "title"    : "Elasticsearch - The definitive guide",
    "authors"  : "Clinton Gormley",
    "started"  : "2013-02-04",
    "pages"    : 230
}'
```

API tour

Update

```
» curl -X PUT localhost:9200/books/book/1 -d '
{
    "title"    : "Elasticsearch - The definitive guide",
    "authors"  : [ "Clinton Gormley", "Zachary Tong"],
    "started"  : "2013-02-04",
    "pages"    : 230
}'
```

API tour

Delete

```
» curl -X DELETE localhost:9200/books/book/1
```

API tour

Get

```
» curl -X GET localhost:9200/books/book/1
```

API tour

Search

```
» curl -X GET localhost:9200/books/_search?q=elasticsearch
```

```
{
  "took" : 2, "timed_out" : false,
  "_shards" : { "total" : 5, "successful" : 5, "failed" : 0 },
  "hits" : {
   "total" : 1, "max_score" : 0.076713204,
   "hits" : [ {
     "_index" : “books", "_type" : “book", "_id" : "1",
     "_score" : 0.076713204, "_source" : {
       "title" : "Elasticsearch - The definitive guide",
       "authors" : [ "Clinton Gormley", "Zachary Tong" ],
       "started" : “2013-02-04", "pages" : 230
     }
   }]
  }
}
```

API tour

Search Query DSL

```
» curl -XGET 'localhost:9200/books/book/_search' -d '{
  "query": {
      "filtered" : {
          "query" : {
              "match": {
                  "text" :  {
                      "query" : "To Be Or Not To Be",
                      "cutoff_frequency" : 0.01
                   }
              }
          },
          "filter" : {
              "range": {
                "price": {
                "gte": 20.0
                "lte": 50.0
          …
          }
}'
```

kibana

Kibana view


Logstash Search
Kibana 3 milestone pre-3
Options
Query Alias
Images
Search
Query
php
png OR gif
css
Chart
Zoom Out | Dynamic (12239) Images (1
nt per 30m | (40354 hits)
800
600
400
200
0
16:00 07/24
20:00 07/24
00:00 07/25
04:00 07/25
08:00 07/25
12:00 07/25
16:00 07/25
20:00 07/25
00:00 07/26
04:00 07/26
08:00 07/26
12:00 07/26
Terms
Term
Count
Action
id 1856
br 1567
pk 1437
ng 1207
bd 1172
ru 1078
jp 959
mx 678
Missing field 9818
Other values 20582
id (1856) br (1567) pk (1437) ng (1207) bd (1172) ru (1078) jp (959) mx (678)
2000
1500
1000
500
0
time must
field : @timestamp
from : "2013-07-24T21:13:04.742Z"
to : "2013-07-26T21:13:04.743Z"
terms mustNot
field : country
value : cn
Toggle
terms mustNot
field : country
value : us
terms mustNot
field : country
value : id

Kibana view


Search
html
php
png
gif
css
Micro Analysis of extension
extension
Action
Count

Kibana view


Twitter Languages
Options
Query
Charts
Trends
-2.02% (user.lang:"es")
2.64% (user.lang:"fr")
-4.81% (user.lang:"pt")
-0.59% (user.lang:"ar")
8.33% (user.lang:"tr")

Programming language support!

- Java
- JavaScript
- Perl
- PHP
- Python
- Ruby
- Scala
- .Net
- Lua
- Clojure
- Erlang
- Go
- Groovy
- Haskell

What is my problem?

Data provides value for businesses

I have so much data or I can make some!

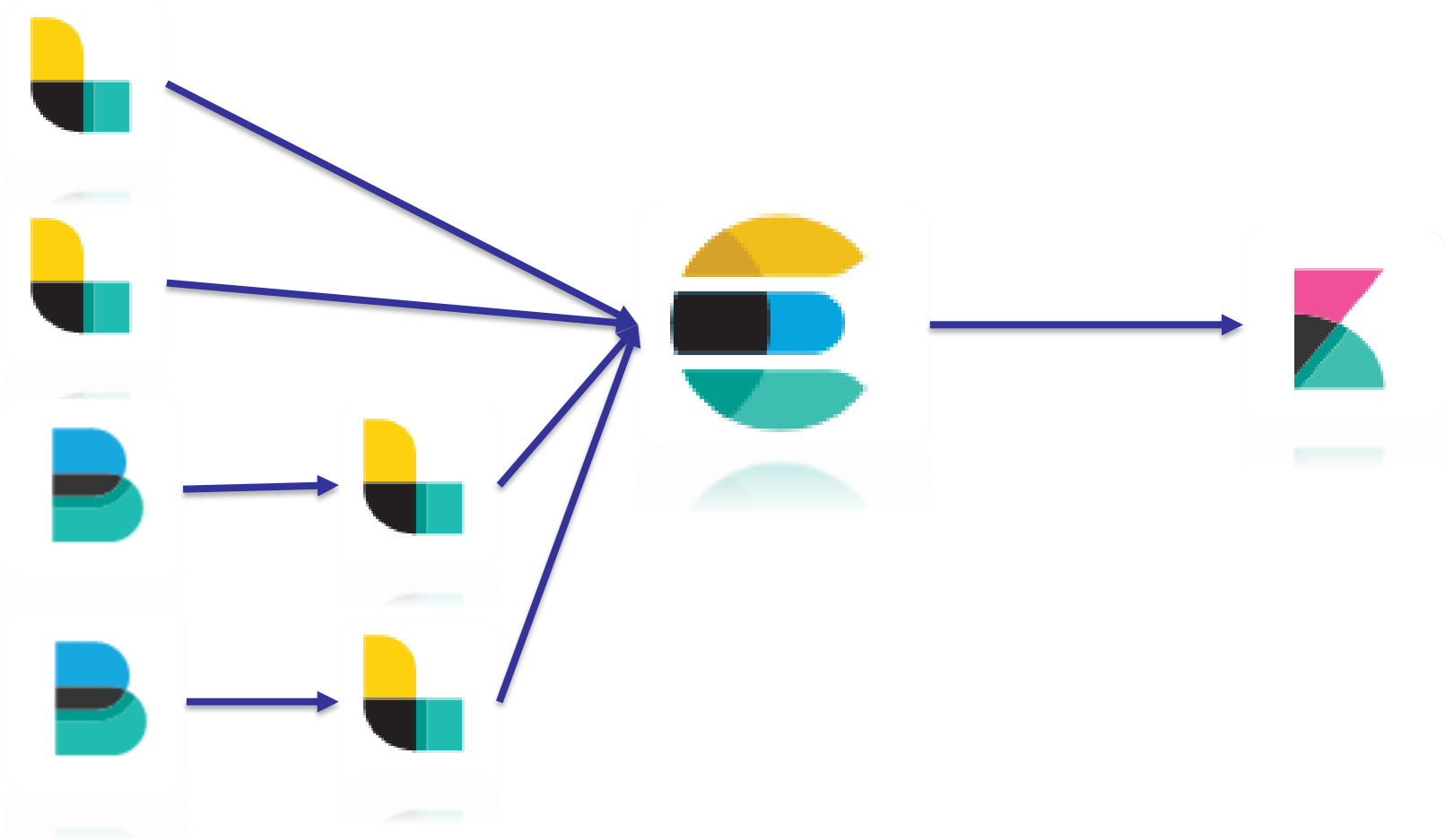
So Let's check what can we do with that data?

Domain data vs Application data

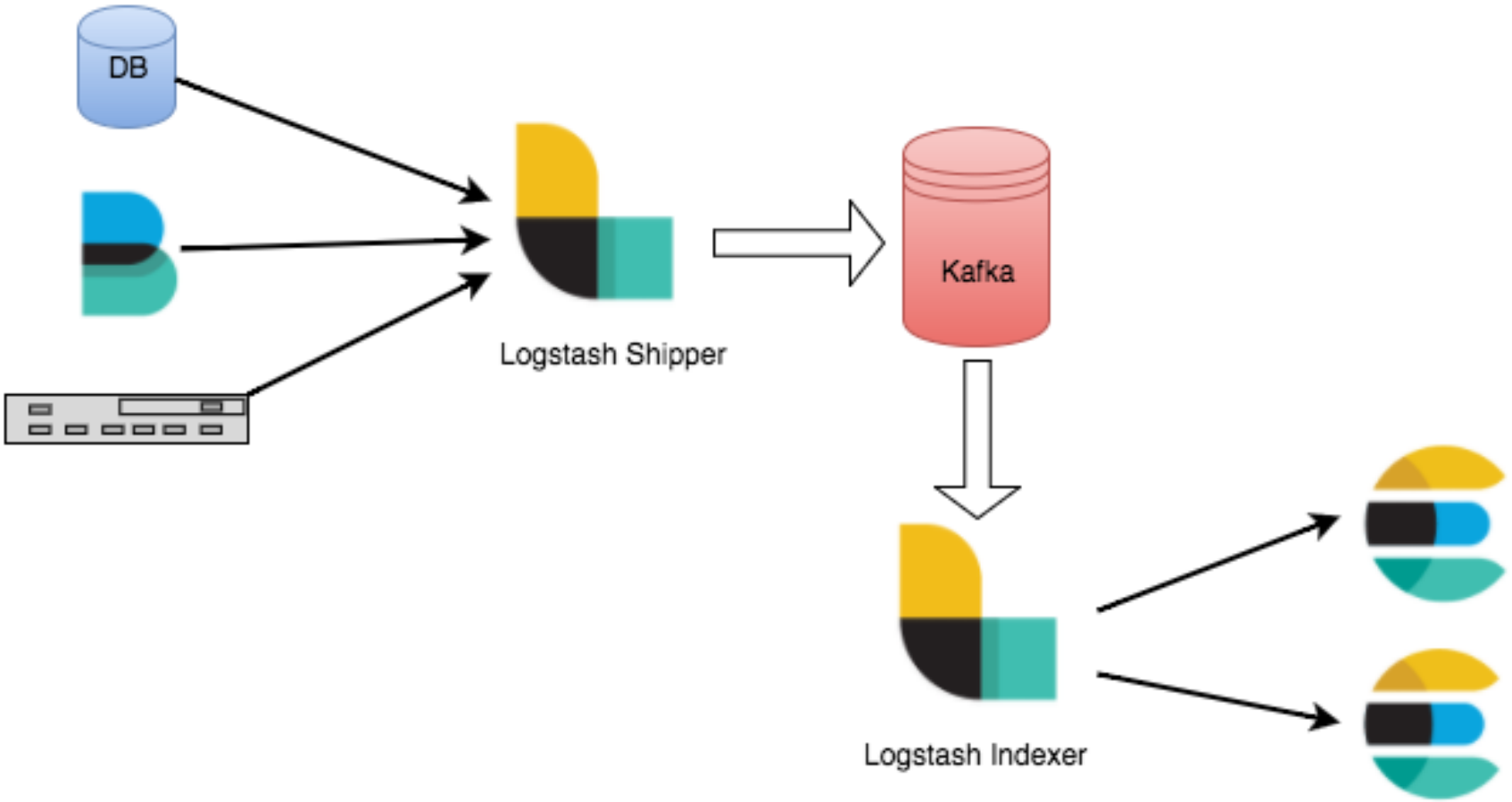
- Internal
  - Orders
  - Products
  - …
- External
  - Social media streams
  - Emails
  - …
- Log files
- Metrics
- Monitoring KPIs
- …

The solution is…

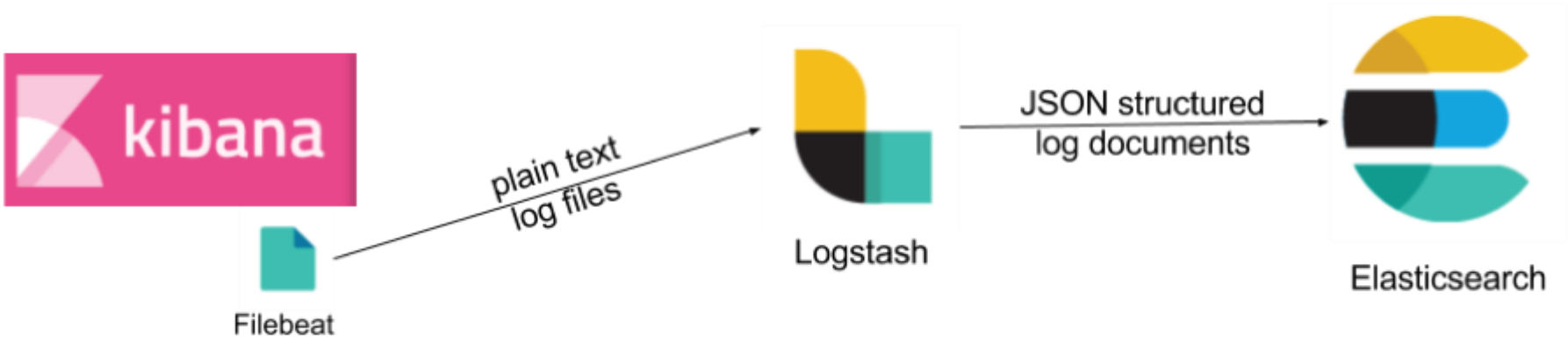
ELK ecosystem



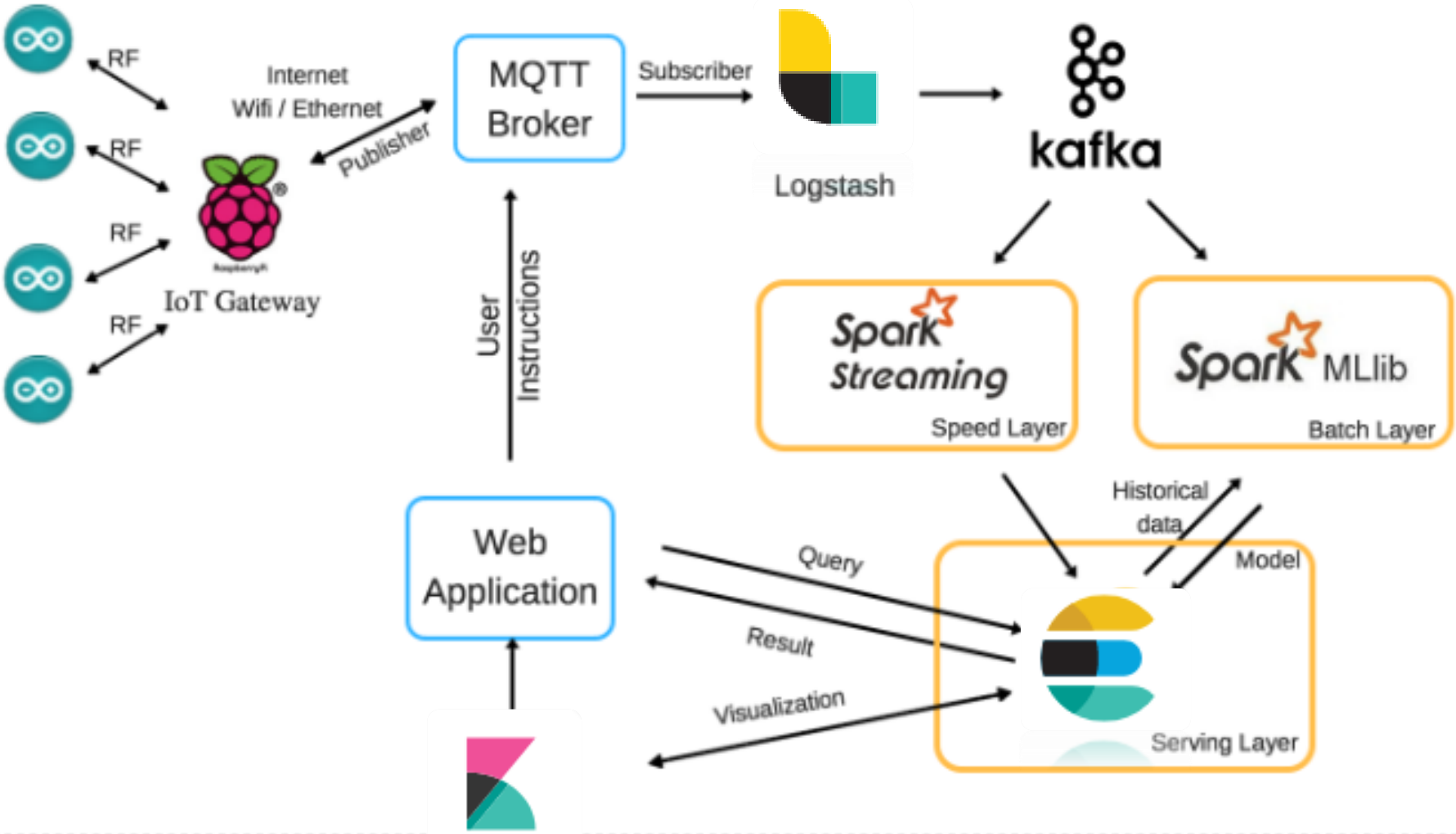
No architectural limitation!



No architectural limitation!



No architectural limitation!



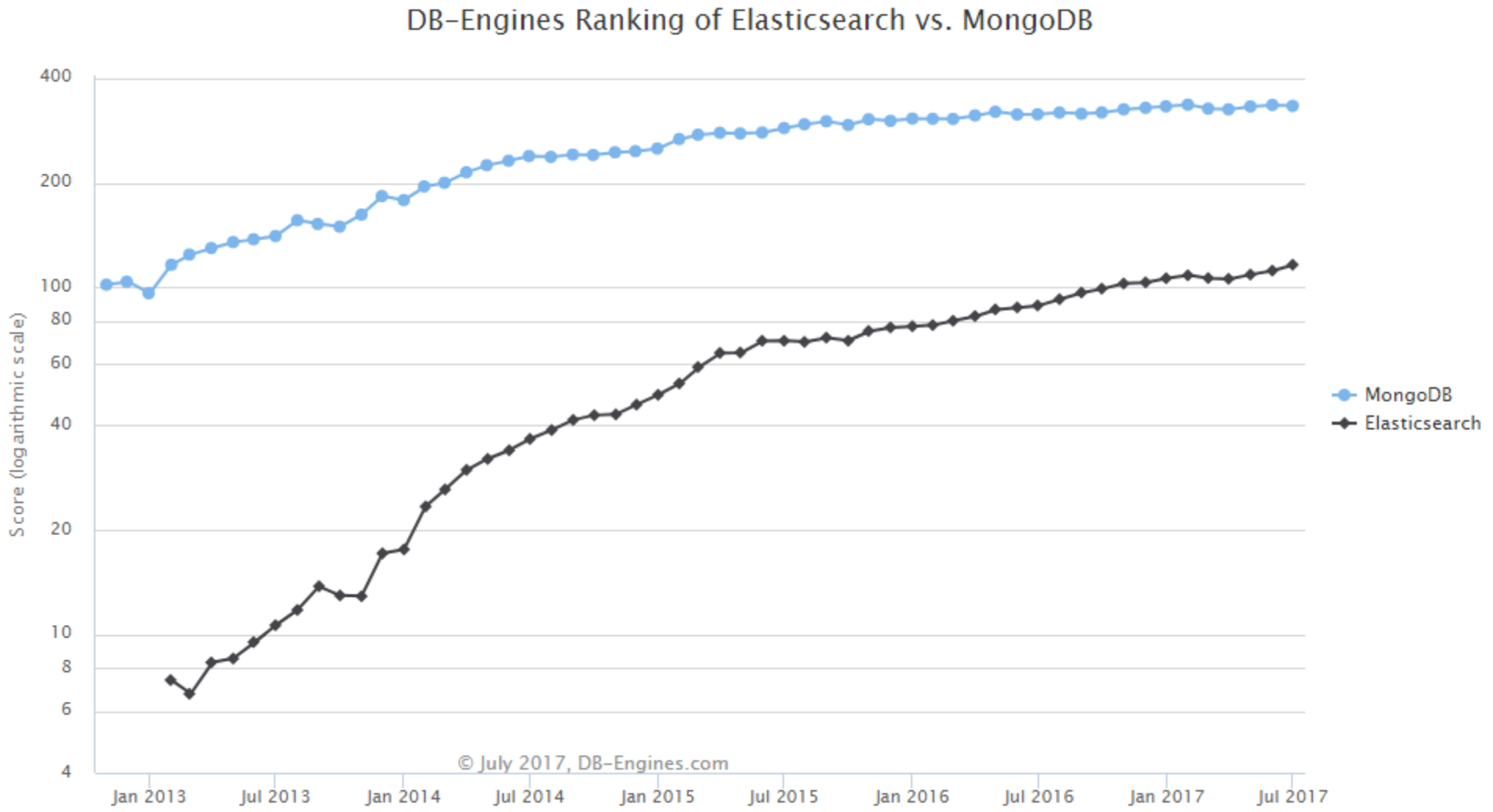
Which technology to choose?

We selected just two out of the box! There are a lot more…

But which one is better?

MongoDB or Elastic search?

According to statistics



According to statistics

Rank Jul 2017	Rank Jun 2017	Rank Jul 2016	DBMS	Database Model	Score Jul 2017	Score Jun 2017	Score Jul 2016
1.	1.	1.	Oracle	Relational DBMS	1374.88	+23.11	-66.65
2.	2.	2.	MySQL	Relational DBMS	1349.11	+3.80	-14.18
3.	3.	3.	Microsoft SQL Server	Relational DBMS	1226.00	+27.03	+33.11
4.	4.	↑ 5.	PostgreSQL	Relational DBMS	369.44	+0.89	+58.28
5.	5.	↓ 4.	MongoDB	Document store	332.77	-2.23	+17.77
6.	6.	6.	DB2	Relational DBMS	191.25	+3.74	+6.17
7.	7.	↑ 8.	Microsoft Access	Relational DBMS	126.13	-0.42	+1.23
8.	8.	↓ 7.	Cassandra	Wide column store	124.12	-0.00	-6.58
9.	9.	↑ 10.	Redis	Key-value store	121.51	+2.63	+13.48
10.	↑ 11.	↑ 11.	Elasticsearch	Search engine	115.98	+4.42	+27.36
11.	↓ 10.	↓ 9.	SQLite	Relational DBMS	113.86	-2.84	+5.33
12.	12.	12.	Teradata	Relational DBMS	78.37	+1.04	+4.43
13.	13.	13.	SAP Adaptive Server	Relational DBMS	66.91	-0.61	-3.82
14.	14.	14.	Solr	Search engine	66.02	+2.41	+1.33
15.	15.	15.	HBase	Wide column store	63.62	+1.75	+10.48
16.	16.	↑ 18.	Splunk	Search engine	60.30	+2.78	+13.65
17.	17.	↓ 16.	FileMaker	Relational DBMS	58.65	+1.57	+7.09
18.	18.	↑ 20.	MariaDB	Relational DBMS	54.36	+1.47	+18.56
19.	19.	19.	SAP HANA	Relational DBMS	47.94	+0.45	+6.14
20.	20.	↓ 17.	Hive	Relational DBMS	46.20	+1.82	-1.34

According to statistics

For more detailed Comparison check this link:

https://db-engines.com/en/system/Elasticsearch%3BMongoDB

Why ELK?

- Actually they are totally different solutions:
  - MongoDB is a general purpose database
  - Elastic search is a distributed text search engine

It's so simple. Don't use them in a situation which they aren't designed for !!

Why ELK?

- ✓ Use mongoDB for your data store but Elastic search for your high performance and customizable full-text-search.
- ✓ Although they have a portion of the other one's capabilities, but they are not the perfect choice for that one!

Why ELK?

- The main idea is that we can decide between a huge number of technologies even when we know the below:
  - What is it designed for?
  - What is the main idea behind the technology?
  - What is my problem?
  - What are my limitations?
  - How can I design the architecture in a way that it solve the solution?
  - …

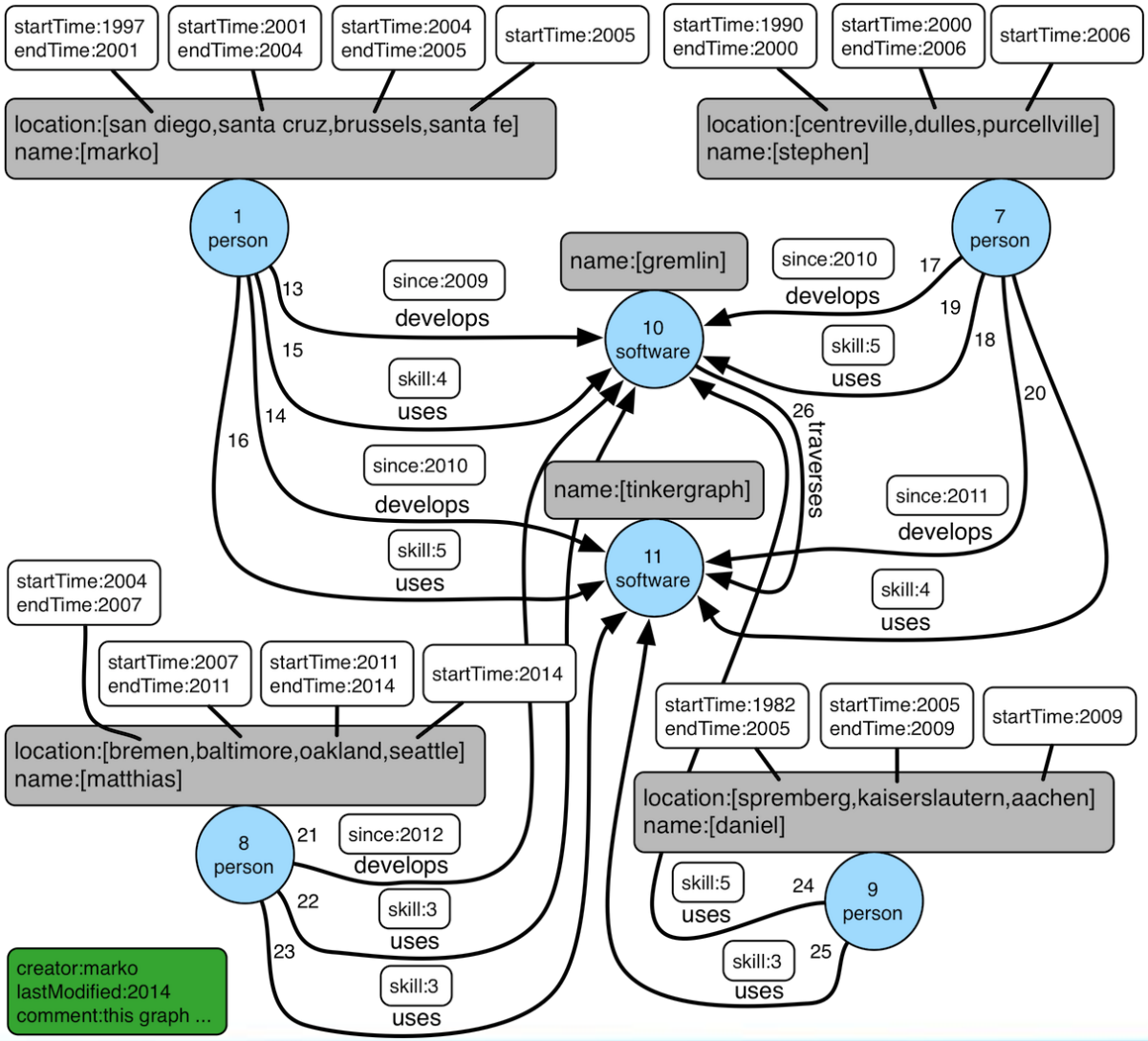
TITAN – Graph Info

- **Specific benefits of TITAN**
  - Open-source and support for very large graphs.
  - Support for many concurrent transactions and operational graph processing.
  - Support for global analytics and batch graph processing through the Hadoop framework.
  - Support full text search for vertices and edges on large graphs.
  - Native support for popular property graph data mode
    - Exposed by Tinkerpop.

TITAN – Graph Info (Cont.)

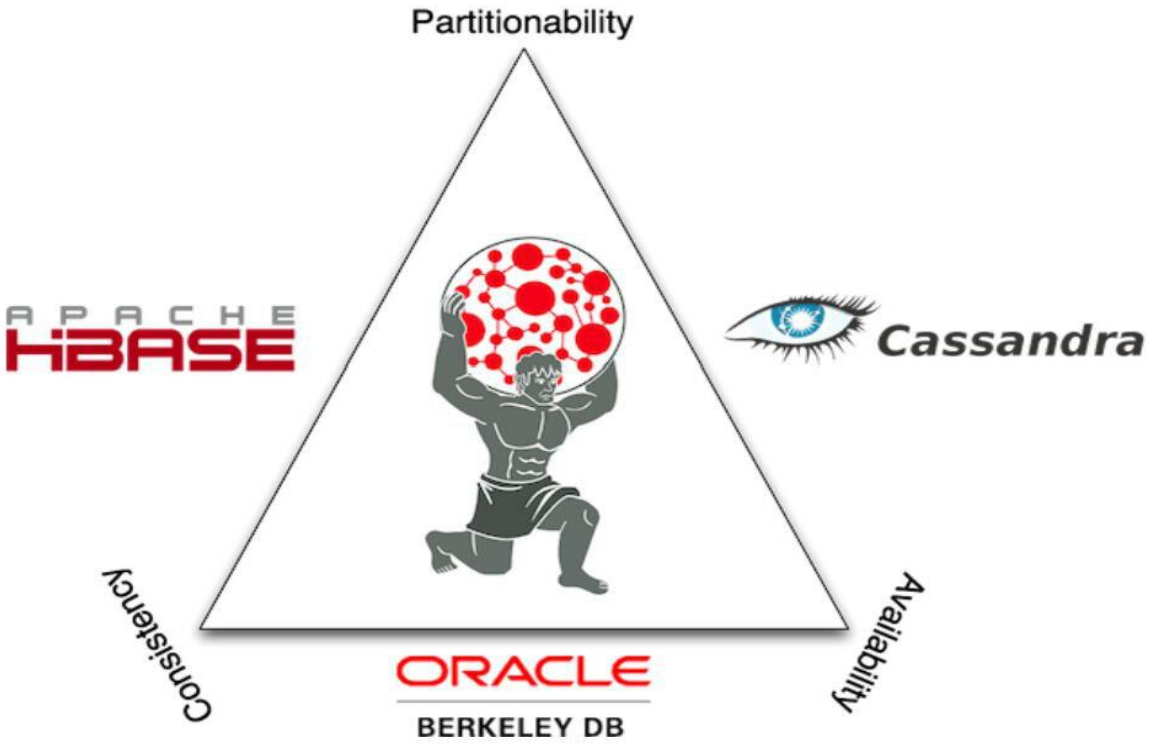
- **Specific benefits of TITAN (Cont.)**
  - Native support for the graph traversal language, the Gremlin.
  - Easy integration with the Gremlin graph server.
  - Numerous graph-level configurations provide knobs for tuning performance.
  - Provide an optimized disk representation to allow for efficient use of storage and speed of access.

TITAN – Graph Info (Cont.)



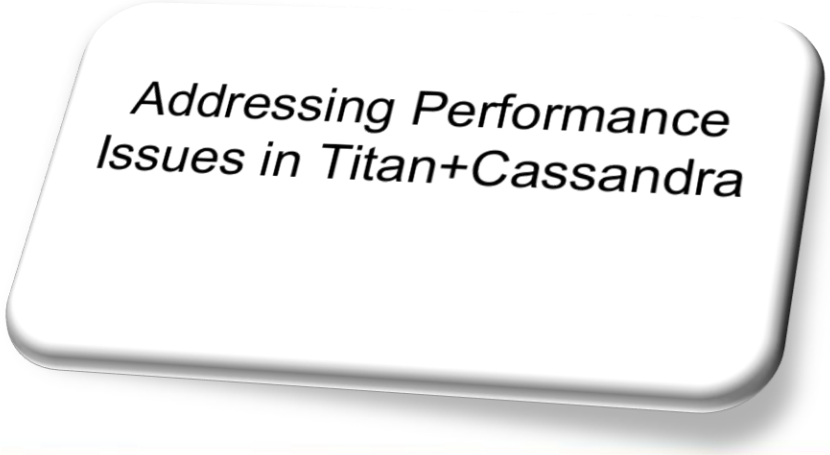
TITAN – Graph Info (Cont.)

- **CAP Theorem : (3 supporting backbends)**
  - C= Consistency, A= Availability, P= Partitionability



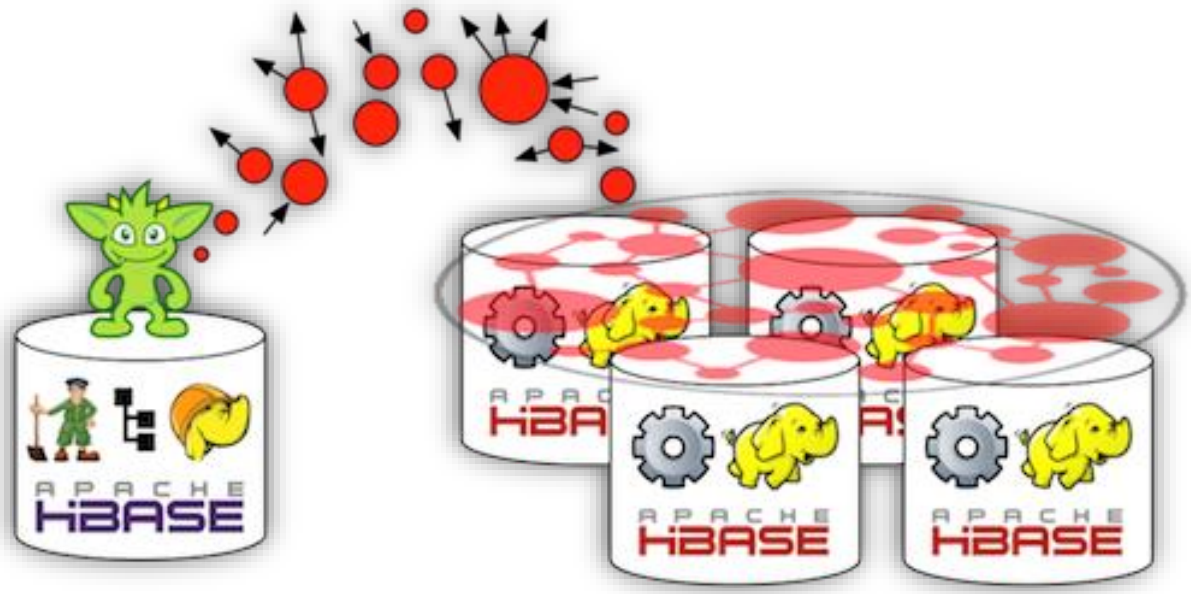
TITAN – Graph Info (Cont.)

- **Benefits of TITAN with Cassandra:**
  - Continuously available with no single point of failure.
  - No read/write bottlenecks to the graph as there isn't master/slave architecture
  - Caching layer ensures that accessed data is available in memory.
  - Increase the size of the cache by adding more machines to the cluster.



TITAN – Graph Info (Cont.)

- **Benefits of TITAN with HBase:**
  - Tight integration with the Hadoop ecosystem.
  - Native support for strong consistency.
  - Convenient base classes for Hadoop Map-Reduce job with HBase



Big Data in the Real World

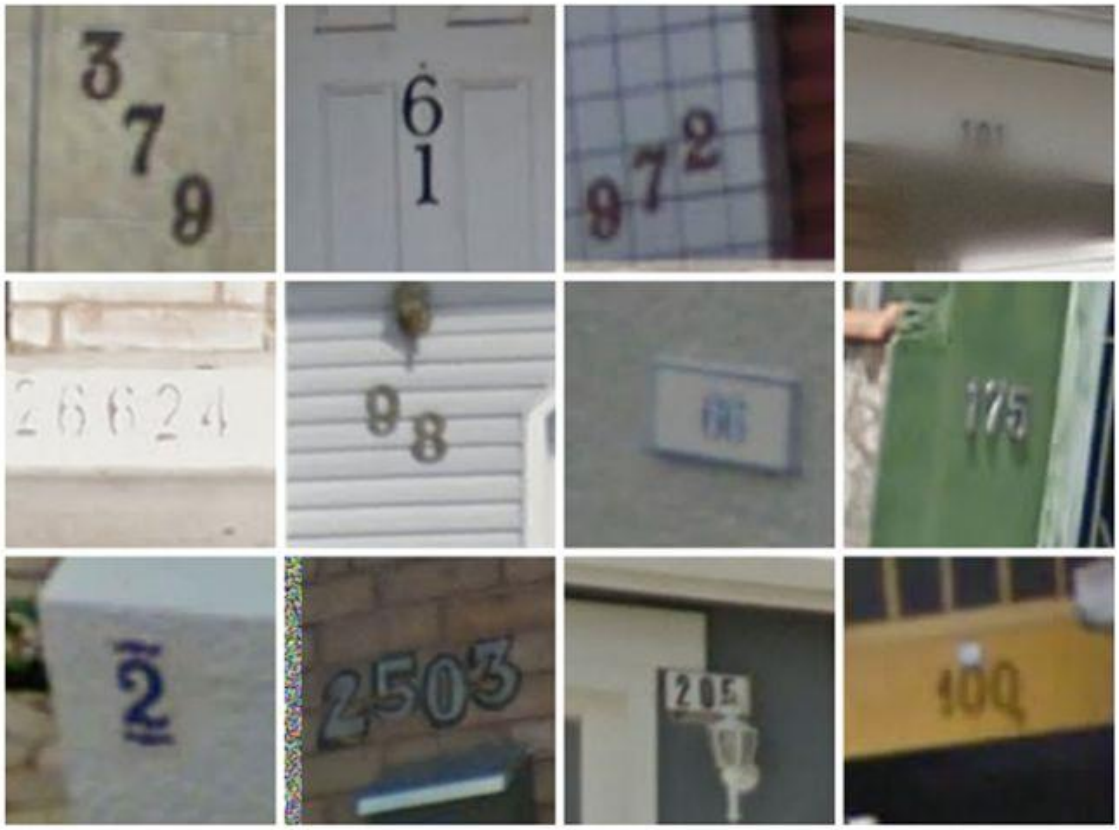
- Social Network Analysis
- Climate data, Large scale health care
- Complex Image Processing
- Personalization (Facebook, Telegram, ….)
- Advertising, Mobile Telecommunication Networks (i.e., 5G),
- E-commerce and E- Banking Applications

Big Data in the Real world (Cont.)

- **Banking Systems; Big Data and Deep Learning**
  - Banknote Authentication and Forgery Detection
  - Financial Fraud Detection
  - Bank Embezzlement & Money Laundering
  - Boost e-commerce Sales
  - Losing From Disgruntled Customers
  - Loan Approval Prediction

Big Data in the real world (Cont.)

- Deep Learning Algorithm Transcribes House Numbers (Google)



Big Data in the real world (Cont.)

- Car Classification using Deep Learning Approach

Contact Info

- Mehdi Habibzadeh (PhD in Computer Science)
  - Telephone and Telegram :
    - +98 912 326 7046
  - Email
    - Nimahm@gmail.com
- Hossein Shemshadi (MSc in IT)
  - Telephone and Telegram :
    - +98 9307513096….
    - @HosseinShemshadi
  - Email
    - Hossein.Shemshadi@gmail.com

Q/A ?

Thanks

Any Question?

☺