JBA

Jargon, Buzzwords & Acronyms

Acknowledgements

Content has been curated and distilled from many different web sources**

* ie shamelessly plagerized, munged together, paraphrased, condensed and oversimplified, possibly to the point of losing original meaning.

* I took little pieces from so many different sources, I didnt track which picture, info and phrasing came from where so here is a lazy blanket acknowledgement of the many untracked sources

The Pipelines

Assorted technologies strung together to do various tasks

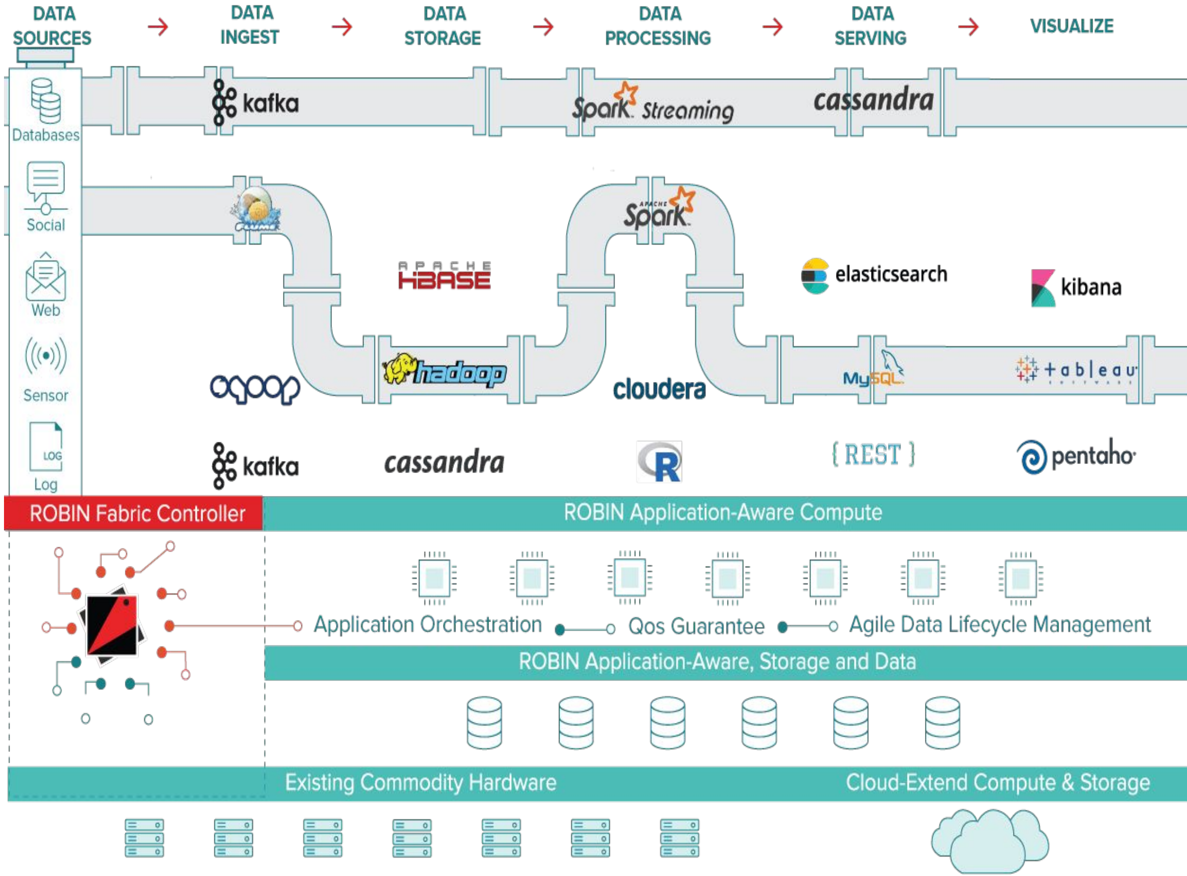
CI Pipeline

Continuous Integration (CI) is a development practice that requires developers to integrate code into a shared repository several times a day. Each check-in is then verified by an automated build, allowing teams to detect problems early. By integrating regularly, you can detect errors quickly, and locate them more easily.

[Big] Data Pipeline

Assorted technologies strung together to do the following:

- Extract data from multiple sources:
  - Eg: web traffic logs, sensor data
- Store and Manage data
  - security, availability, data provenance, data lifecycle
- Analyse data
  - simple statistics, machine learning
- Visualize data



ETL or ETL Pipeline

- ETL is just the beginning of the overall data pipeline
- Extract Transform Load
- Used to describe moving data from one source into another. Esp when loading data into a datastore such as a data warehouse
- Extract: get data out of the source location. Could be static batch of data or a stream on continuous input
- Transform: data into the format required by the destination. Clean formatting, filter, consolidate duplicates,
- Load: data into the destination store
- The E, T and L steps could be done as a Spark job. Eg:

```
#extract
val dataLakeDF = spark.read.parquet("s3a://some-bucket/foo")
val extractDF = dataLakeDF.where(col("mood") === "happy").repartition(10000)
# transform
def model()(df: DataFrame): DataFrame = {df.transform(doTrans1()).transform(doTrans2())}
# load
def exampleWriter()(df: DataFrame): Unit = {val path = "s3a://some-bucket/extracts/bar"
df.write.mode(SaveMode.Overwrite).parquet(path)}
```

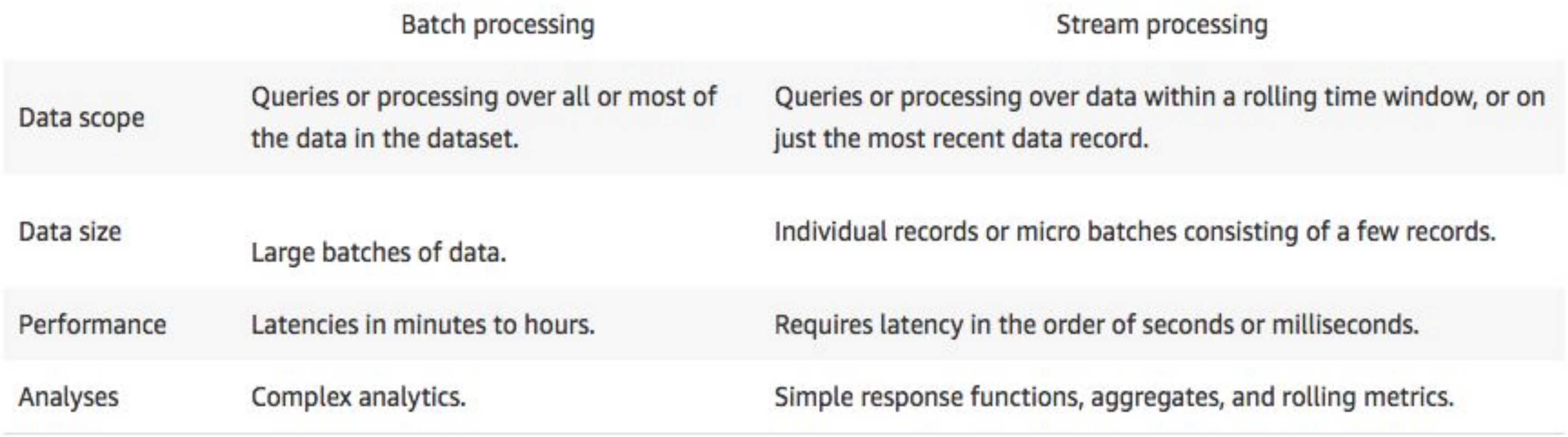
Analytics / Analytics Pipeline

Statistics, machine learning or similar performed on data in order to derive insights or answer business/research questions

Right end of the **data pipeline**

Analytics pipeline is a string of tools and technologies performing a sequence of analytics tasks such as

Batch vs Stream Processing



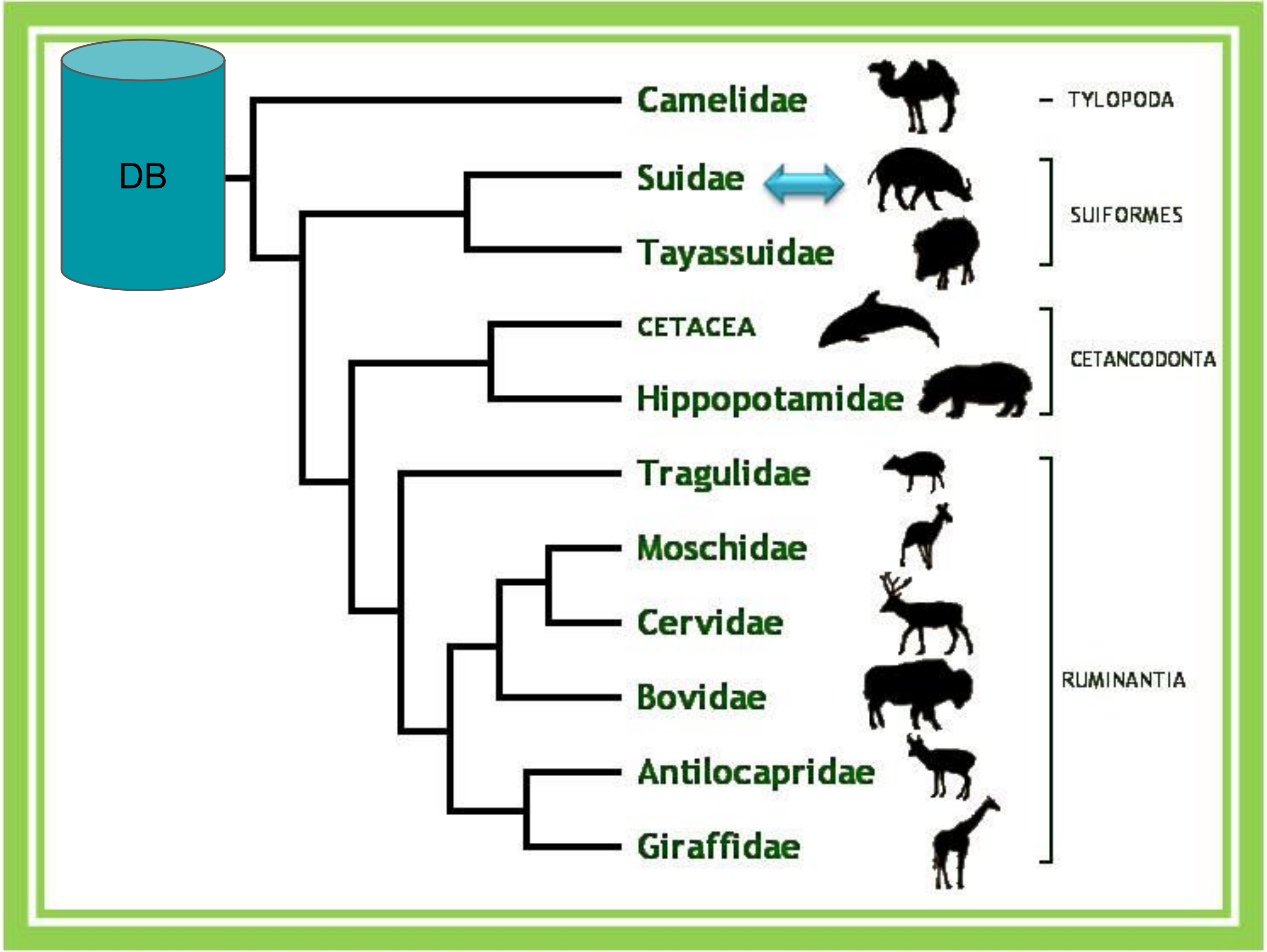
	Batch processing	Stream processing
Data scope	Queries or processing over all or most of the data in the dataset.	Queries or processing over data within a rolling time window, or on just the most recent data record.
Data size	Large batches of data.	Individual records or micro batches consisting of a few records.
Performance	Latencies in minutes to hours.	Requires latency in the order of seconds or milliseconds.
Analyses	Complex analytics.	Simple response functions, aggregates, and rolling metrics.

- deep analysis of big data sets

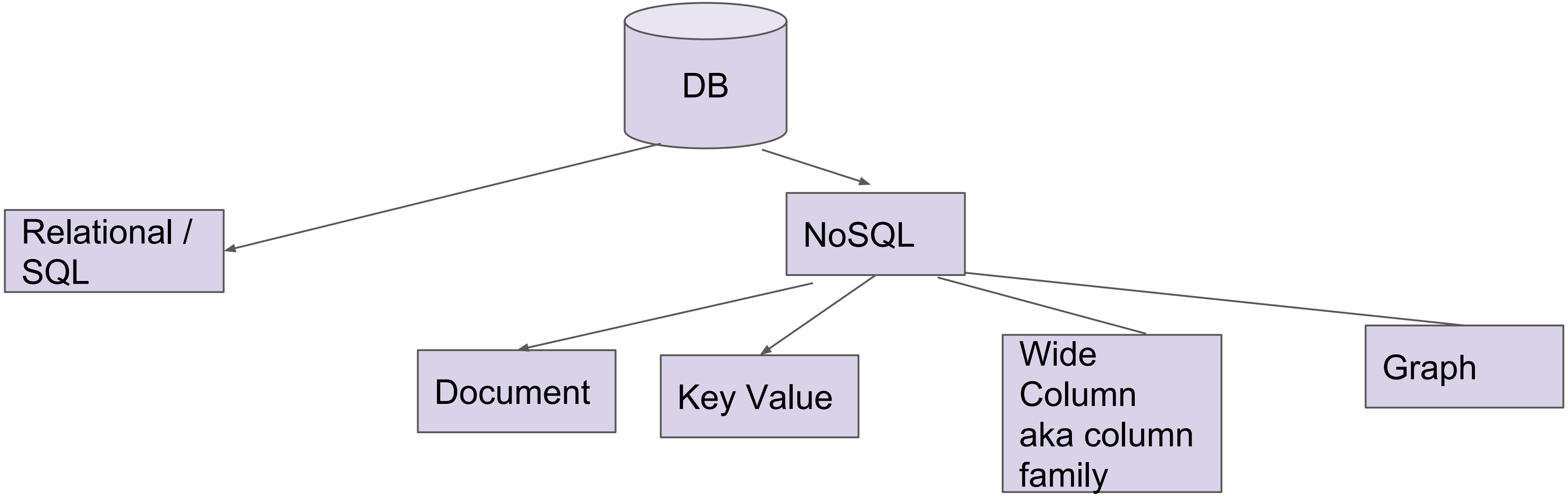
- incrementally updating metrics, reports, and summary statistics in response to each arriving data record
- Often processed in memory before data hits disk

Kafka (Apache Kafka)

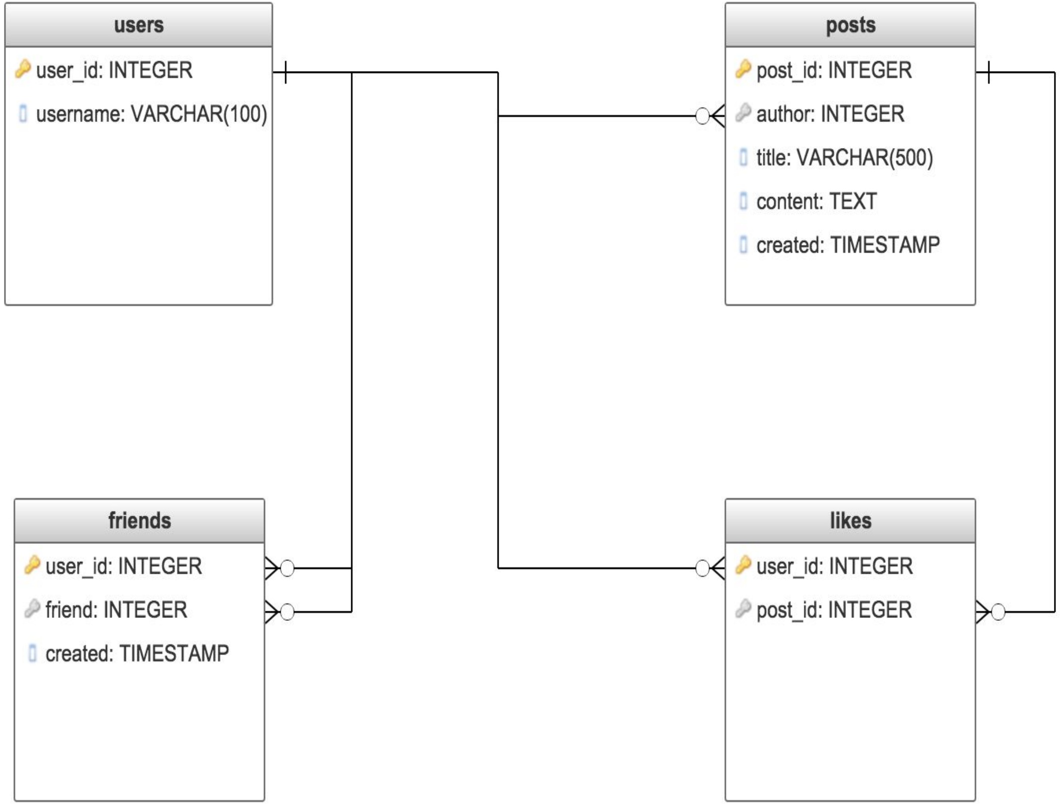
- “Distributed streaming platform”
- It is a way to pass data between different applications, processes or servers and to process streams of data in real time
- It has a publish-subscribe model
- On one-side an application or process publishes data to a “**topic”** using the Kafka **Producer API**
- On the other side an application or process can consume data as and when it is added by listen to the topic using the **Consumer API**
- There is also a **Connector API** that can be used to string together chains of producers and consumers into a pipeline and a **Streams API**
- Kafka runs as cluster of multiple nodes (aka **Kafka brokers**) running Kafka.
- Topic data is partitioned and replicated across multiple brokers


DB
Camelidae
TYLOPODA
Suidae
Tayassuidae
SUIFORMES
CETACEA
Hippopotamidae
CETANCODONTA
Tragulidae
Moschidae
Cervidae
Bovidae
Antilocapridae
Giraffidae
RUMINANTIA

Database taxonomy chart - Sql, NoSql, Graph etc



Relational Database model - tabular



users
- user_id: INTEGER
- username: VARCHAR(100)

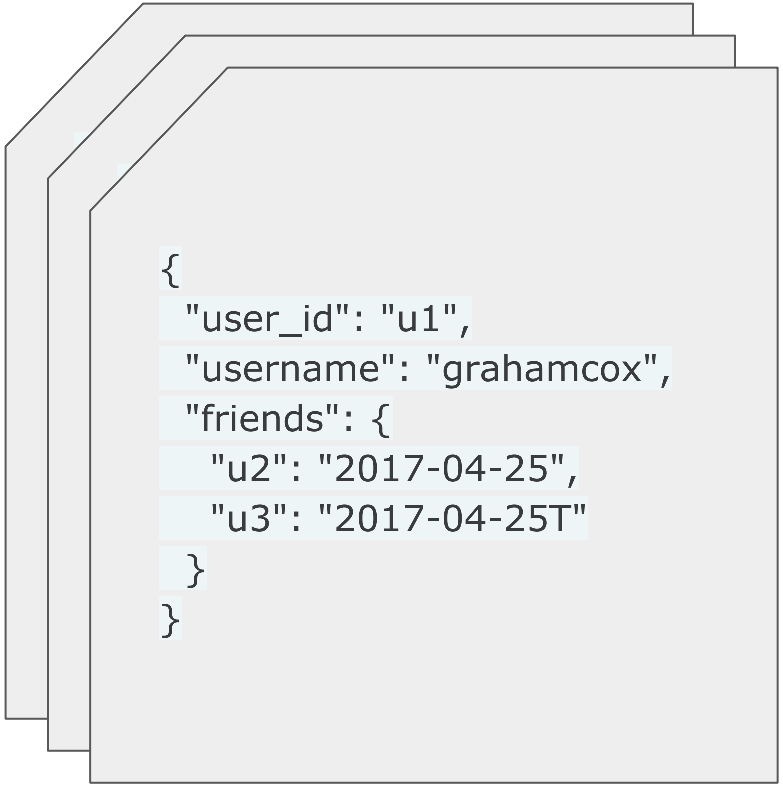
posts
- post_id: INTEGER
- author: INTEGER
- title: VARCHAR(500)
- content: TEXT
- created: TIMESTAMP

friends
- user_id: INTEGER
- friend: INTEGER
- created: TIMESTAMP

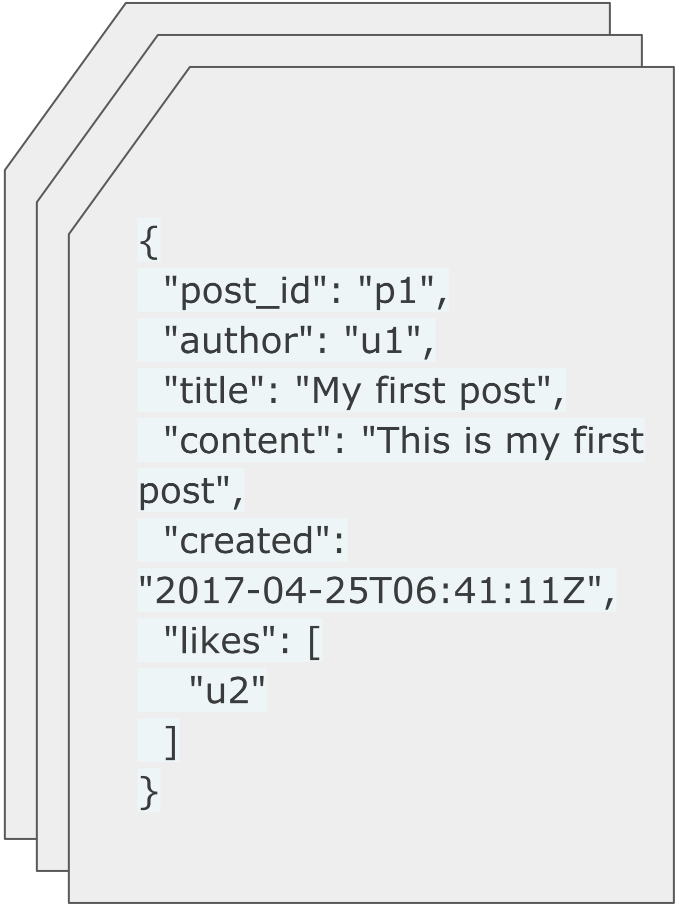
likes
- user_id: INTEGER
- post_id: INTEGER

```
SELECT friends_of_likers.*
FROM posts
JOIN likes ON (posts.post_id = likes.post_id)
JOIN users likers ON (likers.user_id = likes.user
JOIN friends ON (likers.user_id = friends.user_id
JOIN users friends_of_likers ON
(friends_of_likers.user_id = friends.friend)
WHERE posts.author = :me
ORDER BY friends_of_likers.username ASC
```

Document Oriented DB model



```
{
  "user_id": "u1",
  "username": "grahamcox",
  "friends": {
    "u2": "2017-04-25",
    "u3": "2017-04-25T"
  }
}
```

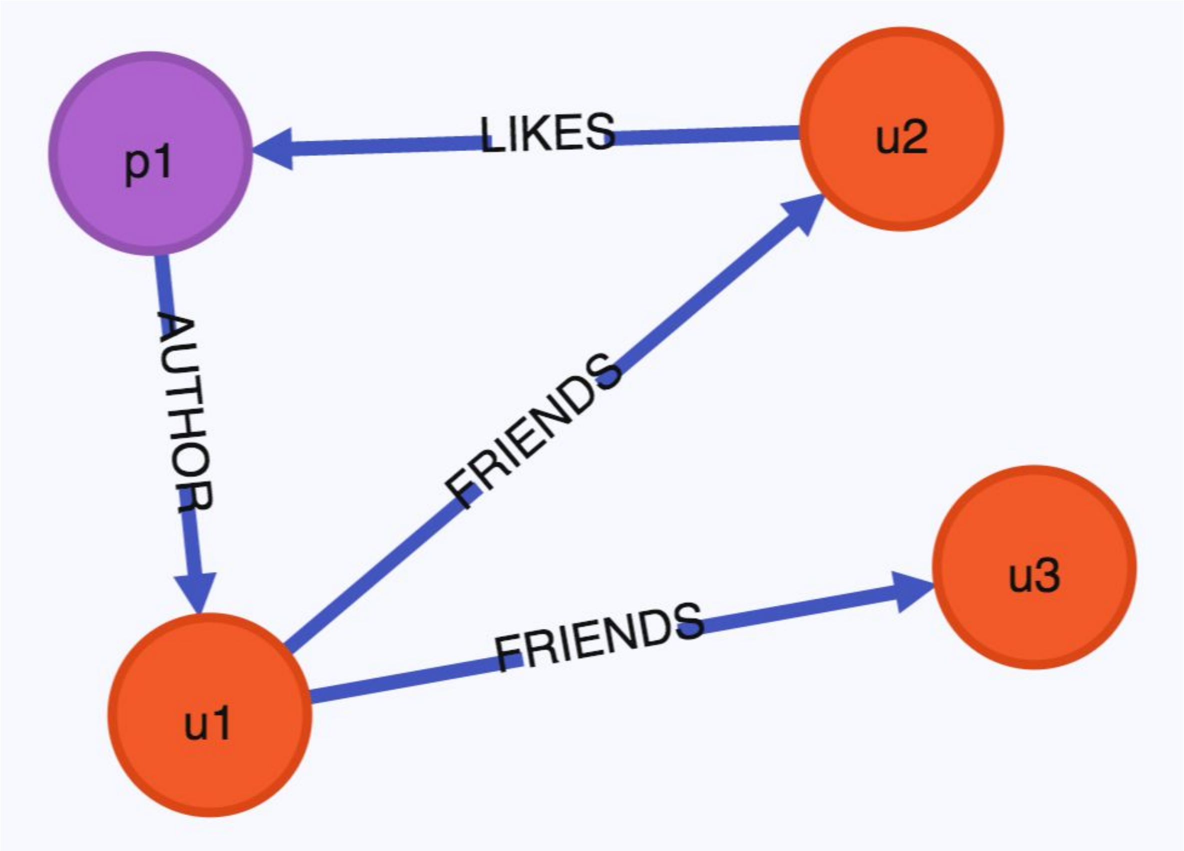


```
{
  "post_id": "p1",
  "author": "u1",
  "title": "My first post",
  "content": "This is my first
post",
  "created":
"2017-04-25T06:41:11Z",
  "likes": [
    "u2"
  ]
}
```

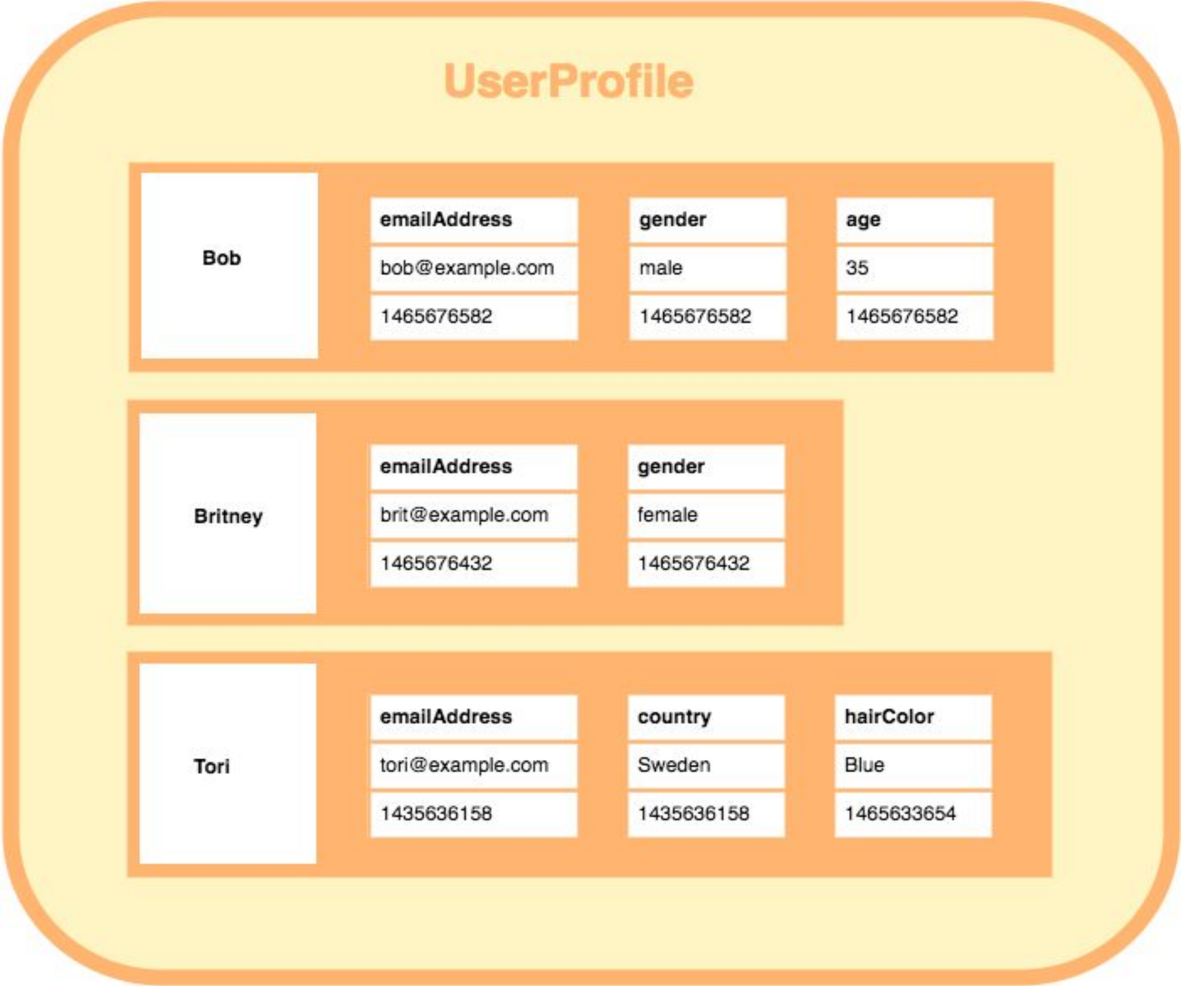
Key Value

Row Id	Key	Value
1	postid	12a34b
2	postauthor	Bjorn
3	posttitle	“Kale breakfast smoothie”
4	postid	25f12a
5	postauthor	Ada
6	posttitle	“One weird trick to …”

Graph DB data model



Wide column store



UserProfile

Bob

emailAddress	gender	age
bob@example.com	male	35
1465676582	1465676582	1465676582

Britney

emailAddress	gender
brit@example.com	female
1465676432	1465676432

Tori

emailAddress	country	hairColor
tori@example.com	Sweden	Blue
1435636158	1435636158	1465633654

NoSQL, (aka non-relational, non-sql, not-only-sql)

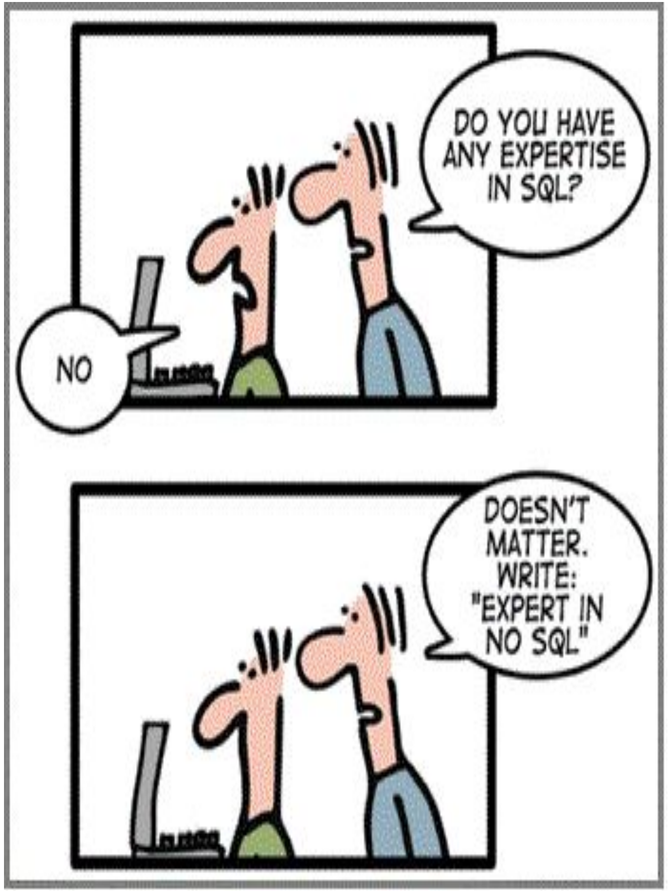
- Compared to relational db, non-sql dbs make it easier to spread data across multiple servers which allows for easier storage of really large datasets

- Often has no fixed schema so dont have to deal with schema versioning etc

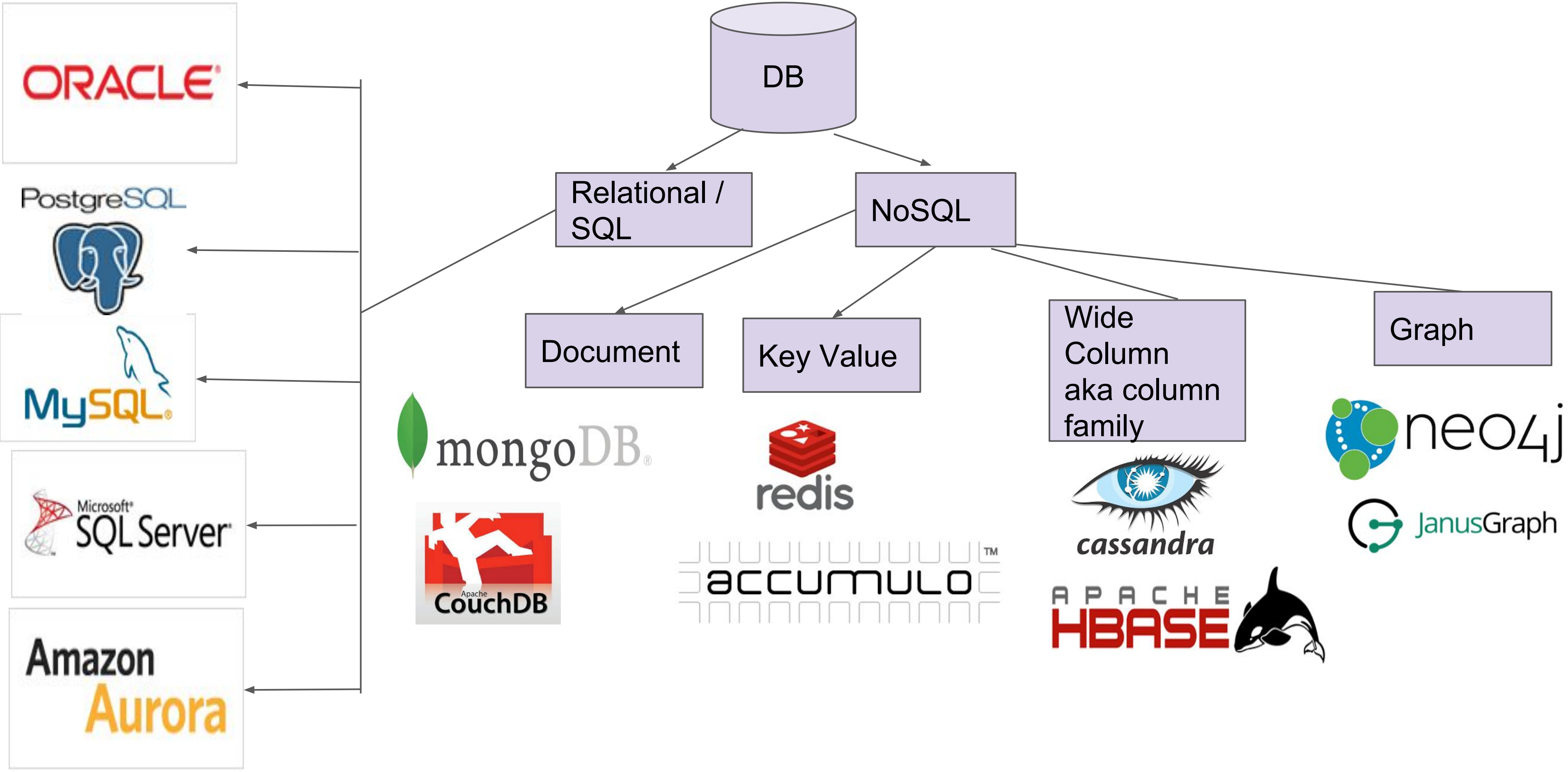
- Different non-tabular data model formats are sometimes particularly suited to certain datasets/ query types. Eg: graph dbs or unstructured blob data

- SQL databases emphasizes on ACID properties (Atomicity, Consistency, Isolation and Durability) whereas the NoSQL database follows the Brewers CAP theorem (Consistency, Availability and Partition tolerance)

- Lots of open source (free) options


DO YOU HAVE ANY EXPERTISE IN SQL?
NO
DOESN'T MATTER. WRITE: "EXPERT IN NO SQL"

Database taxonomy chart - Sql, NoSql, Graph etc



DB

Relational / SQL

NoSQL

Document

Key Value

Wide Column aka column family

Graph

Cassandra (Apache Cassandra)

- A highly performant distributed database
- No sql
- Wide-column store
  - This is a type of no sql DB
  - HBase is another wide column store
- Accumulo is also a distributed, nosql DB
- https://academy.datastax.com/resources/brief-introduction-apache-cassandra

HBase (Apache HBase)

- NoSql database
- Column oriented
- distributed database
- runs on top of Hadoop

Redis

- Database
- NoSql
- Key value store
- In-memory
- Can be used as a database, cache and message broker
- Used by Twitter, Github, Pinterest

Structured, unstructured, semi-structured data

Structured

- Data which can be stored in database SQL in table with rows and columns.
- They have relational key and can be easily mapped into pre-designed fields.

Semi-structured

- Has some structure but the structure does not fit into the relational data model
- Eg: CSV, JSON, XML, NoSQL db data

Unstructured

- Videos, images, text, binary data, sensor data

Data Lake

- A data lake is a storage repository that holds a vast amount of raw data in its native format until it is needed.
- No carefully thought out schema defined ahead of time
- Mixed bag of data types
- Variety of tools used to extract insights from this store

Spark

- a cluster computing framework for parallel processing
- Manages and coordinates execution of tasks across the multiple computers in the cluster
    - Divides and distributes the data used for the work
    - Divides and distributes the work that needs to be done
    - Brings everything back together to get results

Analytics / Analytics Pipeline

Statistics, machine learning or similar performed on data in order to derive insights or answer business/research questions

Right end of the **data pipeline**

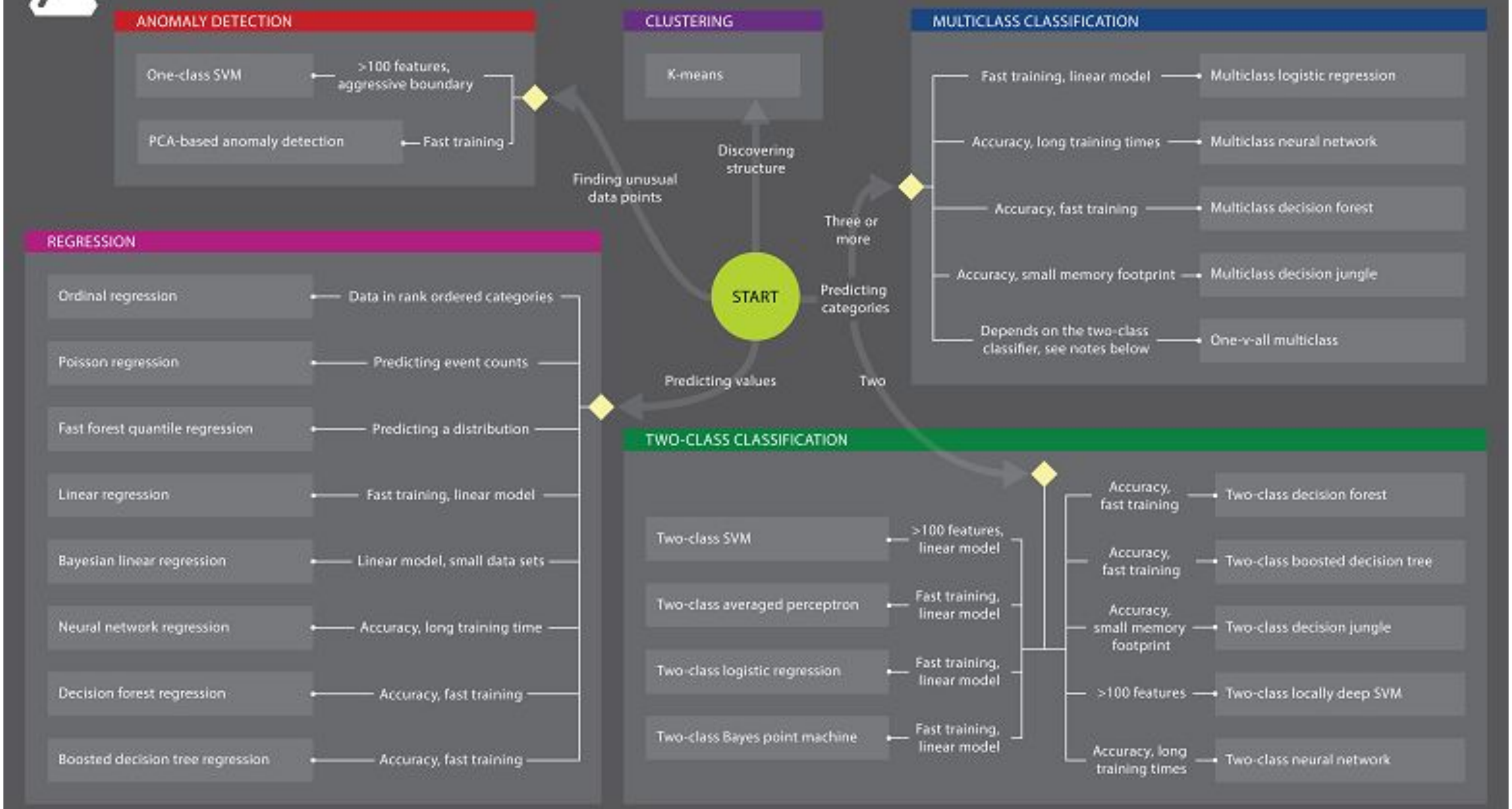
Analytics pipeline is a string of tools and technologies performing a sequence of analytics tasks such as

Machine Learning

- Within the field of data analytics, machine learning is a method used to devise complex models and algorithms that lend themselves to prediction;
- It is an analytics technique - so ML could be one type of analytics performed as part of the data pipeline/ analytics pipeline
- Uses statistical techniques to spot patterns and make inferences
- Eg: Spark Summit keynote speaker from Tesla gave an example of ML vs coded approach to identifying whether a car is parked or not
- Sample capabilities:
  - Predict a customer’s response (favorable/not) to an offer
  - Predict price of a stock
  - Anomaly detection - flag out of the ordinary transactions
  - Of many attributes find the most significant predictors for an outcome, (data preparation)
  - Find items that tend to be bought together and identify their relationship
  - Segment demographic data into clusters

Microsoft Azure Machine Learning: Algorithm Cheat Sheet

This cheat sheet helps you choose the best Azure Machine Learning Studio algorithm for your predictive analytics solution. Your decision is driven by both the nature of your data and the question you're trying to answer.



START

Finding unusual data points

Discovering structure

Three or more

Predicting categories

Two

Predicting values

ANOMALY DETECTION

- One-class SVM – >100 features, aggressive boundary
- PCA-based anomaly detection – Fast training

CLUSTERING

- K-means

MULTICLASS CLASSIFICATION

- Fast training, linear model → Multiclass logistic regression
- Accuracy, long training times → Multiclass neural network
- Accuracy, fast training → Multiclass decision forest
- Accuracy, small memory footprint → Multiclass decision jungle
- Depends on the two-class classifier, see notes below → One-v-all multiclass

REGRESSION

- Ordinal regression – Data in rank ordered categories
- Poisson regression – Predicting event counts
- Fast forest quantile regression – Predicting a distribution
- Linear regression – Fast training, linear model
- Bayesian linear regression – Linear model, small data sets
- Neural network regression – Accuracy, long training time
- Decision forest regression – Accuracy, fast training
- Boosted decision tree regression – Accuracy, fast training

TWO-CLASS CLASSIFICATION

- Two-class SVM – >100 features, linear model
- Two-class averaged perceptron – Fast training, linear model
- Two-class logistic regression – Fast training, linear model
- Two-class Bayes point machine – Fast training, linear model
- Accuracy, fast training → Two-class decision forest
- Accuracy, fast training → Two-class boosted decision tree
- Accuracy, small memory footprint → Two-class decision jungle
- >100 features → Two-class locally deep SVM
- Accuracy, long training times → Two-class neural network

H2O (H2O ai)

- Open source machine learning platform
- Offers implementations of many ML algorithms that you can run on your data
    - Offers a few more algorithms than Spark MLlib
- Can run on a single laptop or on a cluster
- Analogous to:
    - Spark MLlib
    - Apache Mahout

TensorFlow

- Open source library for numerical computation
- Addresses a different need compared to H2O
  - “To sum up: people (mostly) use Tensorflow to implement machine learning stuff (like numpy) and H2O to actually run predefined models, and build pipelines (like scikit-learn).”
- a Python library that allows users to express arbitrary computation as a graph of data flows.
  - Nodes in this graph represent mathematical operations
  - Edges represent data that is communicated from one node to another.
  - Data in TensorFlow are represented as tensors, which are multidimensional arrays.
  - Although this framework for thinking about computation is valuable in many different fields, TensorFlow is primarily used for deep learning in practice and research.

Keras

- high-level neural networks API, written in **Python**
- capable of running on top of TensorFlow, CNTK, or Theano

Data Provenance (Data lineage)

- The two terms are sometimes used interchangeable but have some subtle differences
- **data provenance** includes only high level view of the system for business users, so they can roughly navigate where their data come from. It's provided by variety of modeling tools or just simple custom tables and charts.
- **Data lineage** is a more specific term and includes two sides - business (data) lineage and technical (data) lineage. Business lineage pictures data flows on a business-term level and it's provided by solutions like Collibra, Alation and many others. Technical data lineage is created from actual technical metadata and tracks data flows on the lowest level - actual tables, scripts and statements. Technical data lineage is being provided by solutions such as MANTA or Informatica Metadata Manager.

GPU - Graphics Processing Unit

CPU

- Single to few cores
- Optimized for executing tasks serially
  - Have to wait for one calculation to finish before doing the other

GPU

- Thousands of cores
- Optimized for executing tasks in parallel
- Designed to manipulate matrices of pixels in parallel
  - This is a similar computation to the matrix math used for ML/AI
  - Deep learning runs

Probably should stop here

Remaining slides digress from the data processing theme a bit

Containers - docker containers, lxc containers

- A way to run software in (semi) isolation from it’s surroundings on a given host
- Container images are a way to package software executables with everything needed to run it like code, runtime, system tools, system libraries, & settings
- Because containers are so self-contained you can plop them onto any host and you are guaranteed that you application will run the same way it was tested - no install time/ run time problems due to differences in system package versions etc

Kubernetes

- Basically Kubernetes is a distributed system that runs programs (well, containers) on computers. You tell it what to run, and it schedules it onto your machines.
- https://jvns.ca/blog/2017/06/04/learning-about-kubernetes/

Mesos (Apache Mesos) and Mesosphere

- Cluster manager
- Manages workloads in a cluster - scheduling, distributing
- Opensource code contributed to Apache foundation by the company Mesosphere
- Mesosphere sells a customized distribution of mesos called Mesosphere
  - Kind of like Cloudera and HortonWorks are customized distributions of Apache Hadoop