

ETL

Extract, transform, and load Data

Dennis Wilson

dennis.wilson@isae-superaero.fr

FSD312 Data Engineering

A DAY IN DATA

The exponential growth of data is undisputed, but the numbers behind this explosion – fuelled by internet of things and the use of connected devices – are hard to comprehend, particularly when looked at in the context of one day

500m
tweets are sent every day
Twitter



4PB
of data created by Facebook, including

350m photos
100m hours of video watch time
Facebook Research

320bn

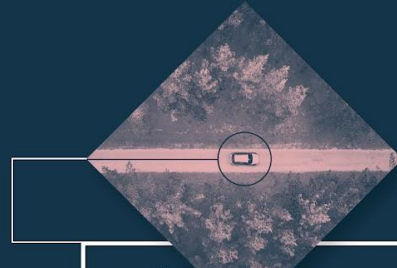
emails to be sent each day by 2021

306bn

emails to be sent each day by 2020

294bn
billion emails are sent
Radical Group

Radical Group



4TB
of data produced by a connected car
Intel

65bn
messages sent over WhatsApp and two billion minutes of voice and video calls made
Facebook



463EB

of data will be created every day by 2025
IDC

95m
photos and videos are shared on Instagram
Instagram Business



3.9bn
people use emails



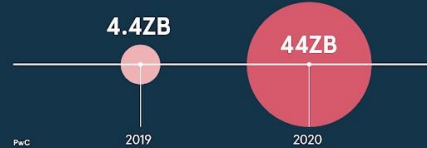
Searches made a day **5bn**
Searches made a day from Google **3.5bn**
Smart Insights



28PB
to be generated from wearable devices by 2020
Statista



ACCUMULATED DIGITAL UNIVERSE OF DATA



DEMYSIFYING DATA UNITS

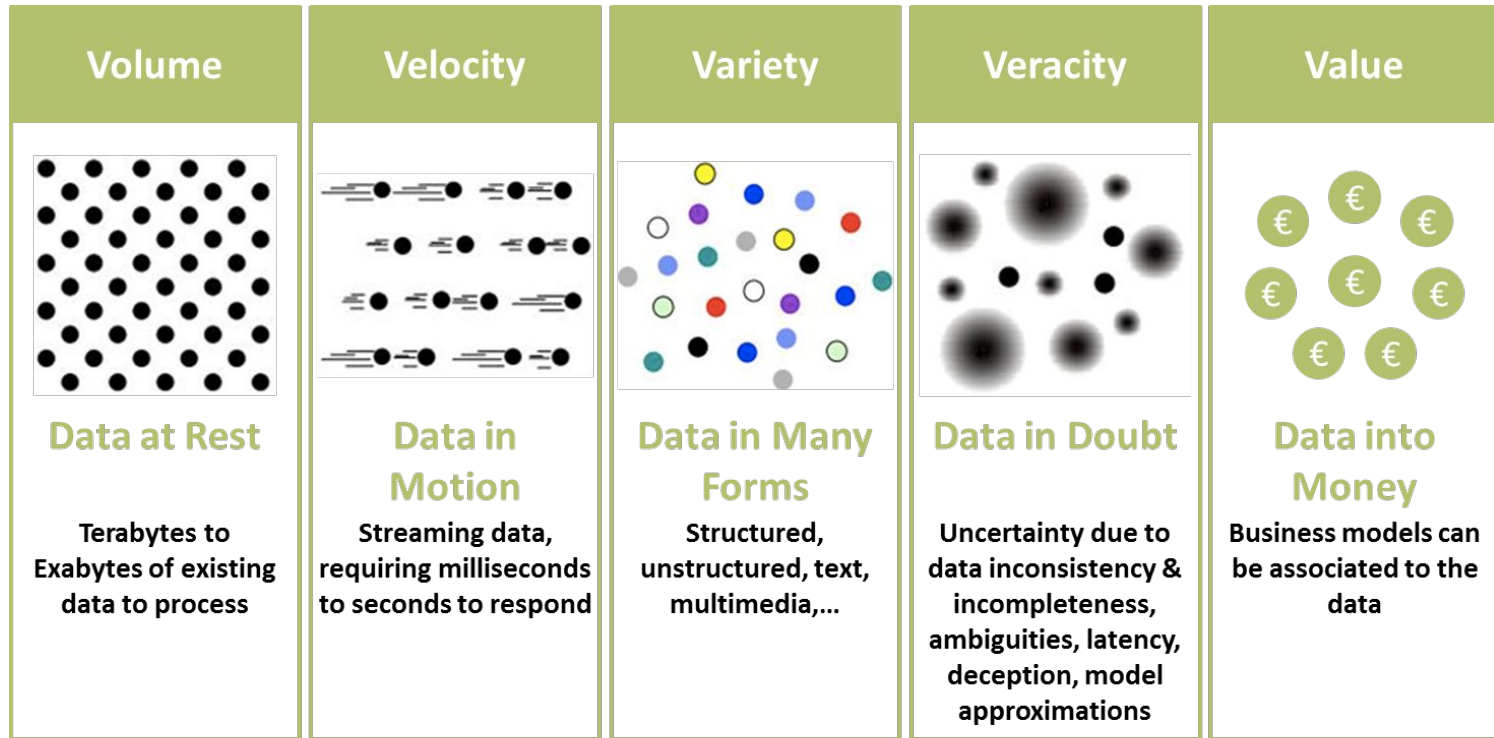
From the more familiar 'bit' or 'megabyte', larger units of measurement are more frequently being used to explain the masses of data

Unit	Value	Size
b bit	0 or 1	1/8 of a byte
B byte	8 bits	1 byte
KB kilobyte	1,000 bytes	1,000 bytes
MB megabyte	1,000 ² bytes	1,000,000 bytes
GB gigabyte	1,000 ³ bytes	1,000,000,000 bytes
TB terabyte	1,000 ⁴ bytes	1,000,000,000,000 bytes
PB petabyte	1,000 ⁵ bytes	1,000,000,000,000,000 bytes
EB exabyte	1,000 ⁶ bytes	1,000,000,000,000,000,000 bytes
ZB zettabyte	1,000 ⁷ bytes	1,000,000,000,000,000,000,000 bytes
YB yottabyte	1,000 ⁸ bytes	1,000,000,000,000,000,000,000,000 bytes

*A lowercase "b" is used as an abbreviation for bits, while an uppercase "B" represents bytes.

5 V's of Big Data

1. Volume
2. Velocity
3. Variety
4. Veracity
5. Value



Adapted by a post of Michael Walker on 28 November 2012

Extract, transform, and load

The ETL Process Explained



Extract

Retrieves and verifies data from various sources



Transform

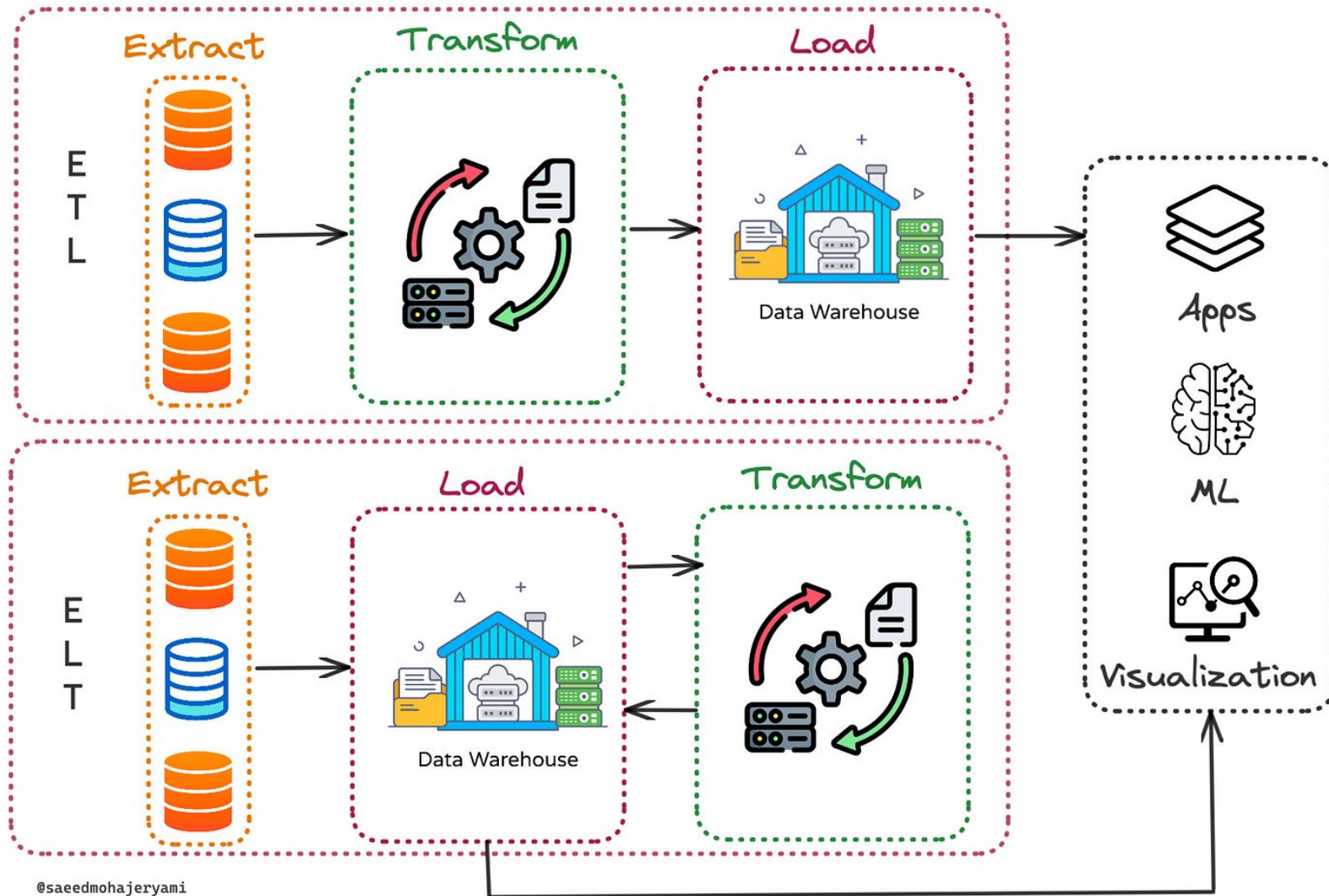
Processes and organizes extracted data so it is usable



Load

Moves transformed data to a data repository

Extract, load, and transform



Extraction

Get data from:

- A database
- A file system
- API: Application Programming

Interface

- Web scraping

Check data sources:

- Is the format correct?
- Did the data arrive when expected?
- Did all data sources return data?
- Do the data make sense?
- Is the data complete, part of a batch, or part of a stream?

Transformation

- Transform data between formats
- Validate and clean data
- ETL: performed through scripts on temporary data, before loading data into long-term storage
- ELT: performed on stored data by scripts in the data platform

Transformation examples:

- Selecting data features
- Validating data
- Replacing missing values
- Encoding free-form values
- Calculating a new data value
- Joining data from multiple sources
- Deduplicating data
- Transposing or pivoting data

Loading

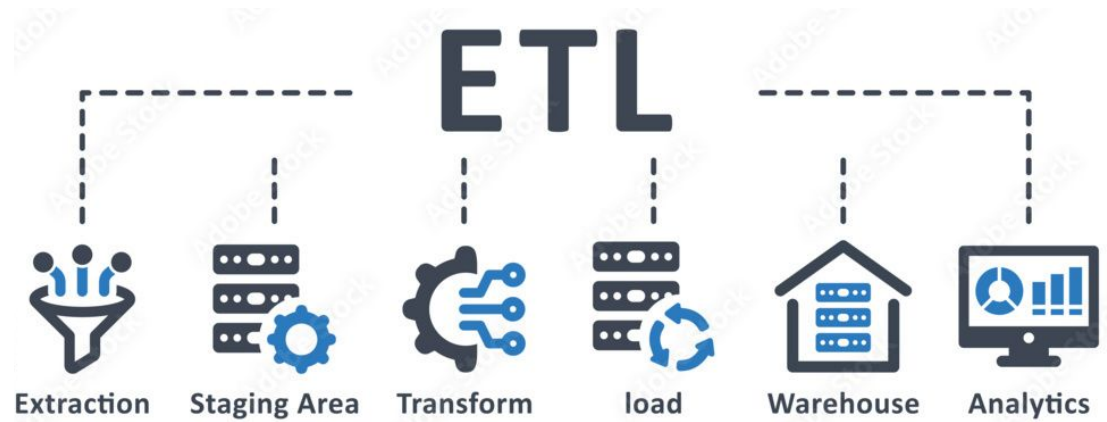
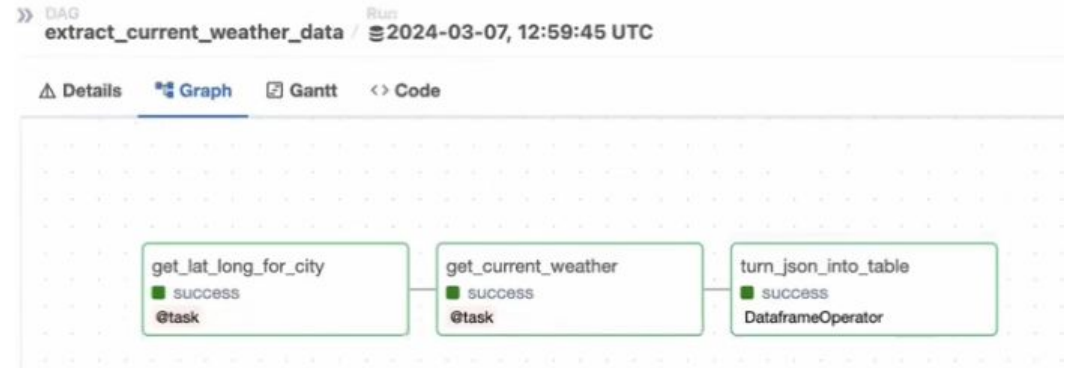
- Store recovered data
- Storage structures:
 - DBMS
 - Data warehouse
 - Data lake
- Allow analysis on stored data

Data warehouse examples:




- Amazon Redshift
- Google BigQuery
- IBM Db2 Warehouse
- Microsoft Azure Synapse
- Oracle Autonomous Data Warehouse
- Snowflake
- Teradata Vantage

ETL tools

- Scripts
- Python
- Apache Airflow
- AWS Glue
- Google Dataflow
- Microsoft SQL Server Integration Services (SSIS)
- Airbyte
- Stitch



Storage structures

<i>factor-bytes.com</i>	 Database	 Data Lake	 Data Warehouse
Data	Structured	Raw & Unstructured	Structured
Data Processing	Schema-on-write	Schema-on-read	Schema-on-write
Scalability	Varies	High	High
Cost	0 - \$	\$\$	\$\$\$
Users	Anyone	Data Scientists	Business Users
Use Cases	Real-time data processing, high transactional throughput, strong data consistency	Exploratory analysis, machine learning, data mining, or data science research	Reporting, analytics, and business intelligence

Database Management Systems (DBMS)

Système de gestion de base de données (SGBD)

Relational



PostgreSQL

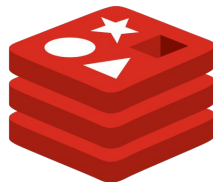
NoSQL (Not only SQL)



mongoDB®



CouchDB
relax



redis

NewSQL

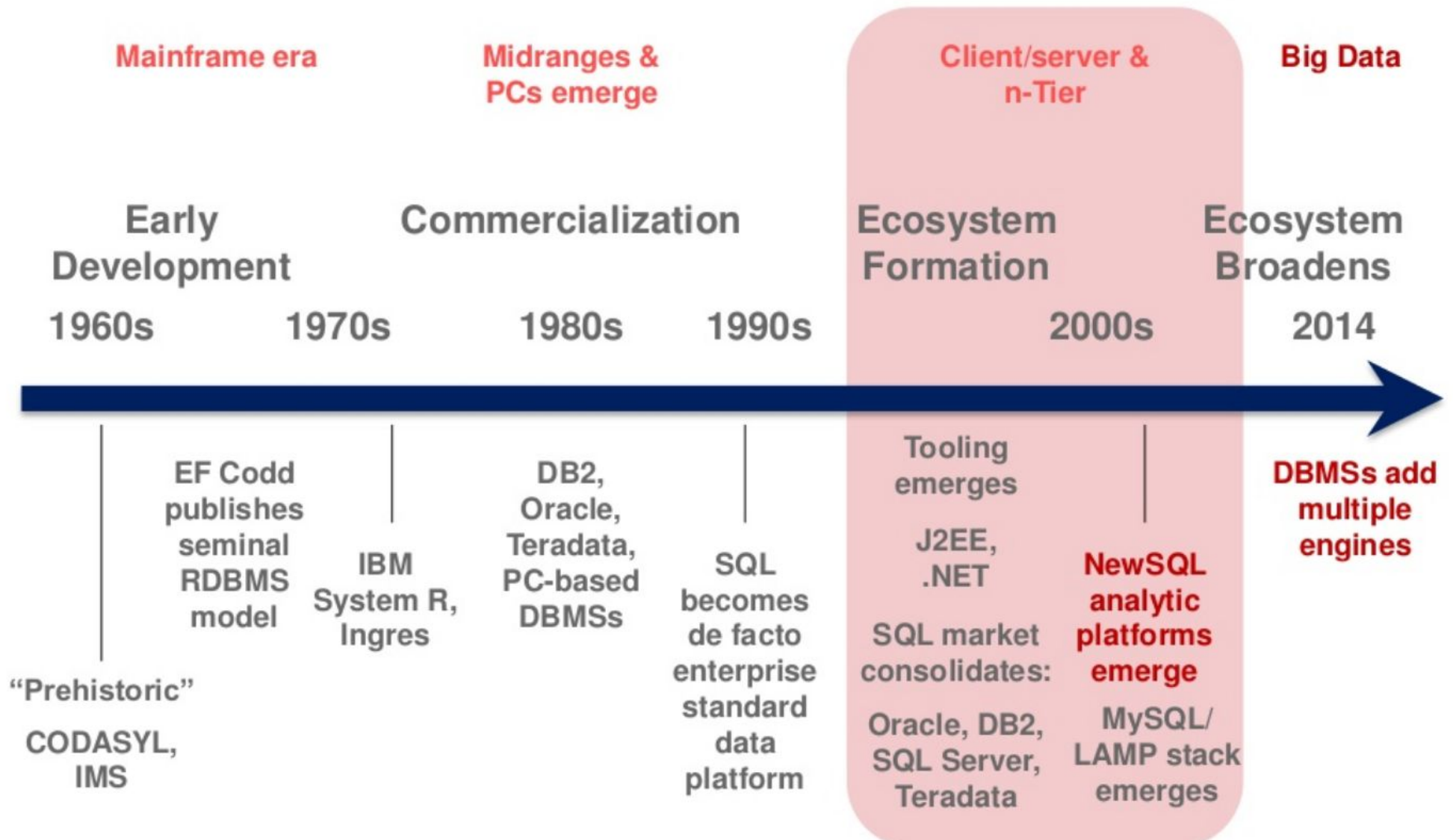


Cockroach DB



Apache
Trafodion

DBMS history



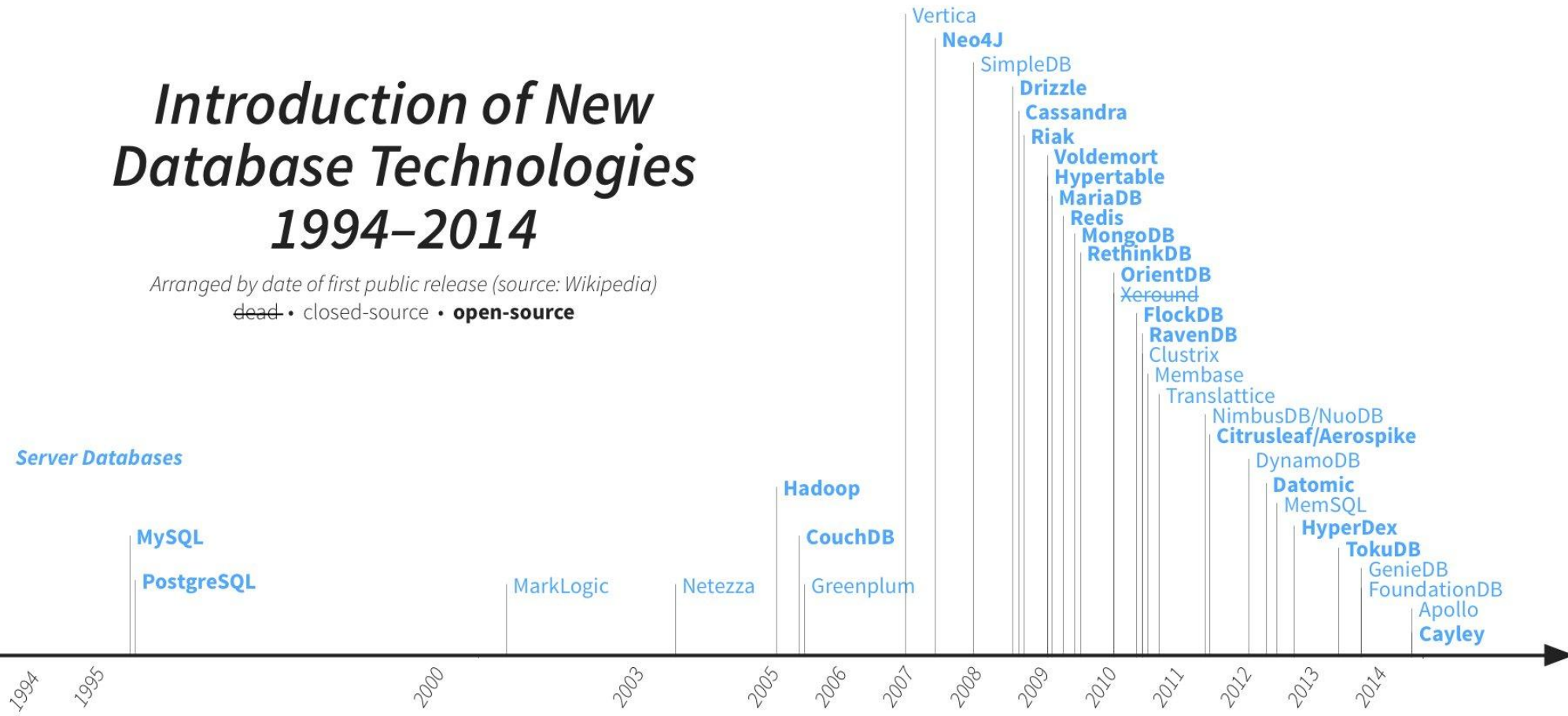
Recent DBMS history

Introduction of New Database Technologies 1994–2014

Arranged by date of first public release (source: Wikipedia)

~~dead~~ • closed-source • **open-source**

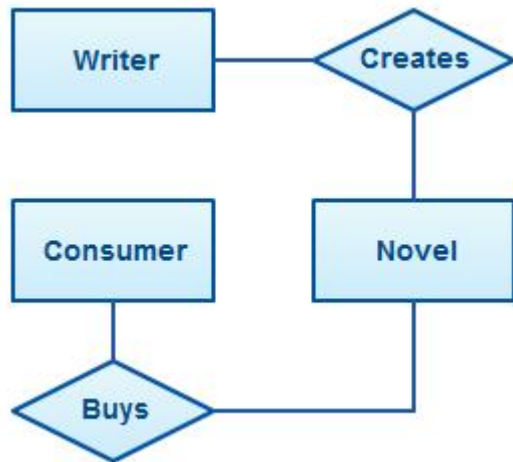
Server Databases



RDBMS : relations

Writer

ID	First Name	Last Name	Affiliation	Age
382	George	Martin	ASOIAF	72



Creates

Writer_ID	Novel_ID
382	9772

Novel

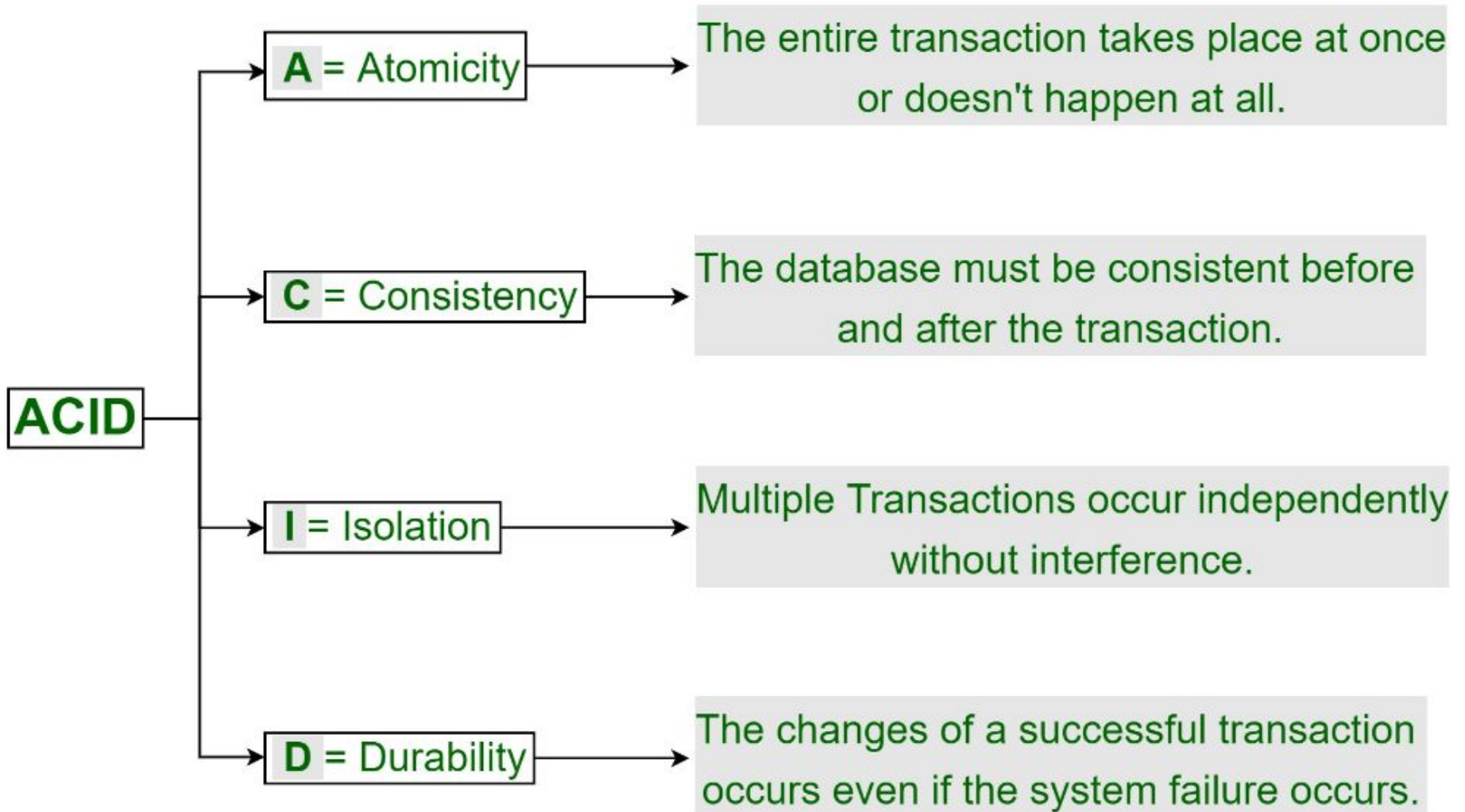
ID	Title	Publication Year	Genre
9772	A Game of Thrones	1996	Fantasy



PostgreSQL

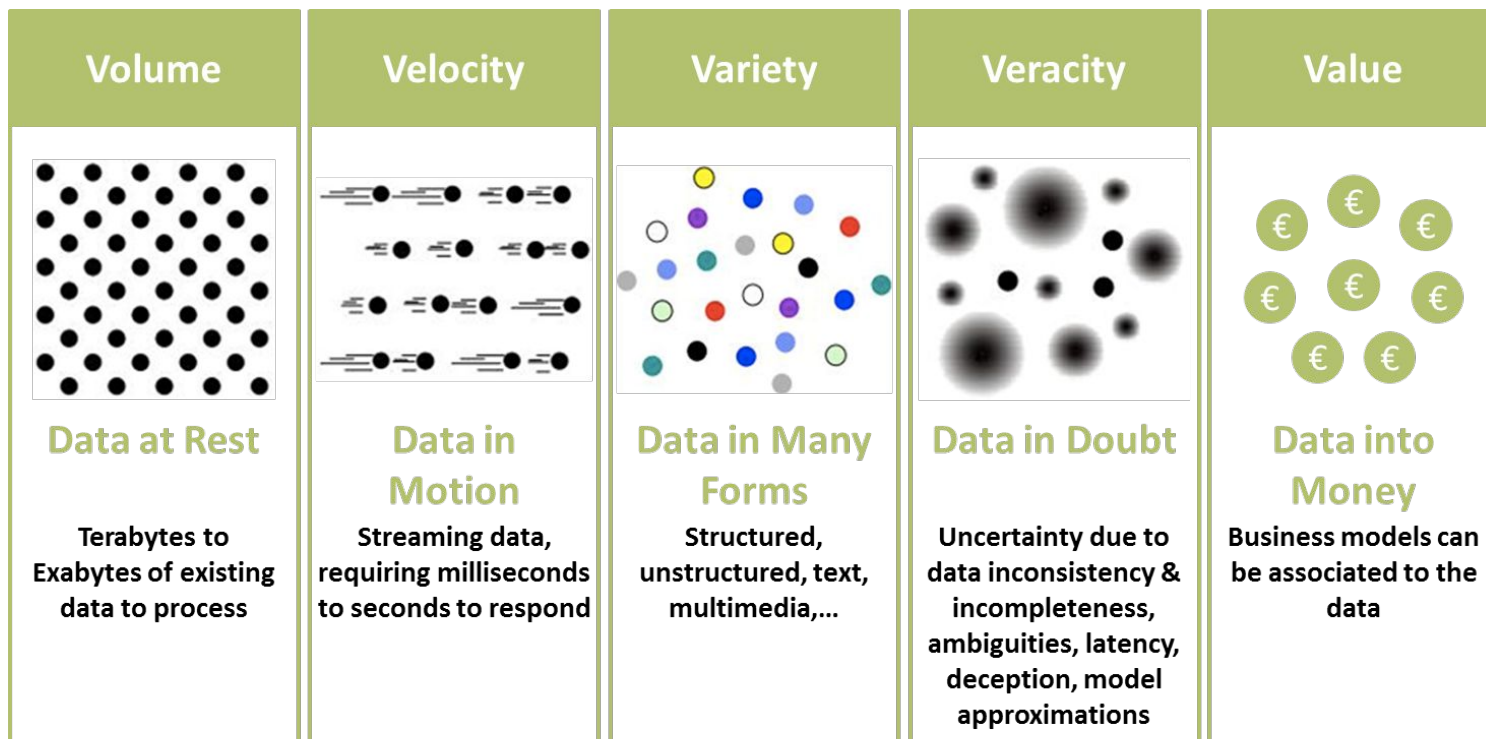


ACID



Limits of RDBMS

1. RDBMS can not handle 'Data Variety'
2. Difficult to change tables and relationships
3. Structural limitations
4. Strict application of ACID principles

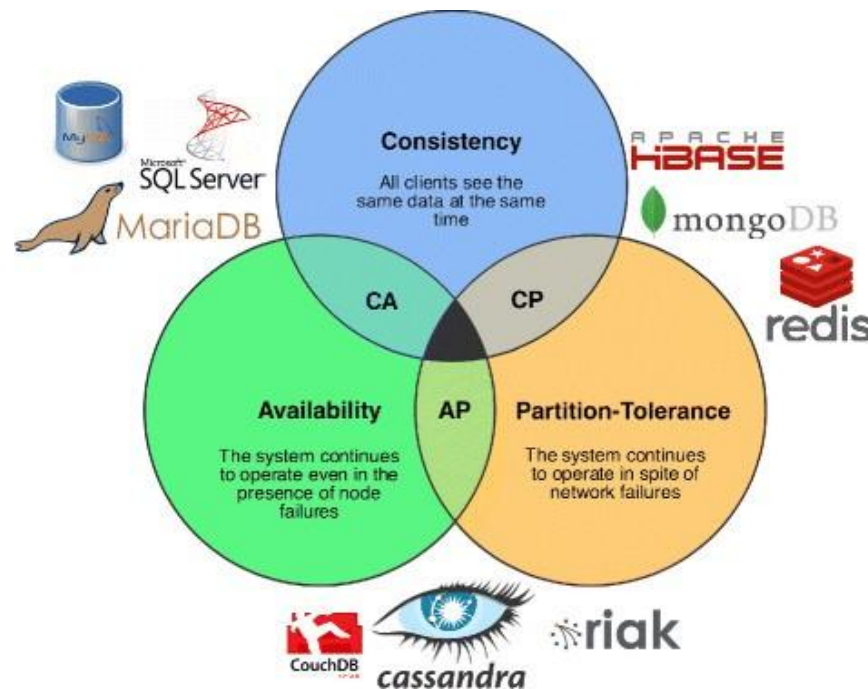


Adapted by a post of Michael Walker on 28 November 2012

Brewer's theorem (CAP)

It is impossible for a distributed data store to simultaneously provide more than two out of the following three guarantees:

- **Consistency:** Every read receives the most recent write or an error
- **Availability:** Every request receives a (non-error) response, without the guarantee that it contains the most recent write
- **Partition tolerance:** The system continues to operate despite an arbitrary number of messages being dropped (or delayed) by the network between nodes



BASE

- **Basically Available:** basic reading and writing operations are available as much as possible (using all nodes of a database cluster), but without any kind of consistency guarantees (the write may not persist after conflicts are reconciled, the read may not get the latest write)
- **Soft state:** without consistency guarantees, after some amount of time, we only have some probability of knowing the state, since it may not yet have converged
- **Eventually consistent:** If the system is functioning and we wait long enough after any given set of inputs, we will eventually be able to know what the state of the database is, and so any further reads will be consistent with our expectations

Main objectives for NoSQL databases

NoSQL (Not Only SQL)

Characteristics of NoSQL:

- Schema free
- Eventually consistent (as in the BASE property)
- Replication of data stores to avoid Single Point of Failure.
- Can handle Data variety and huge amounts of data.

Key value Stores — Riak, Voldemort, and Redis

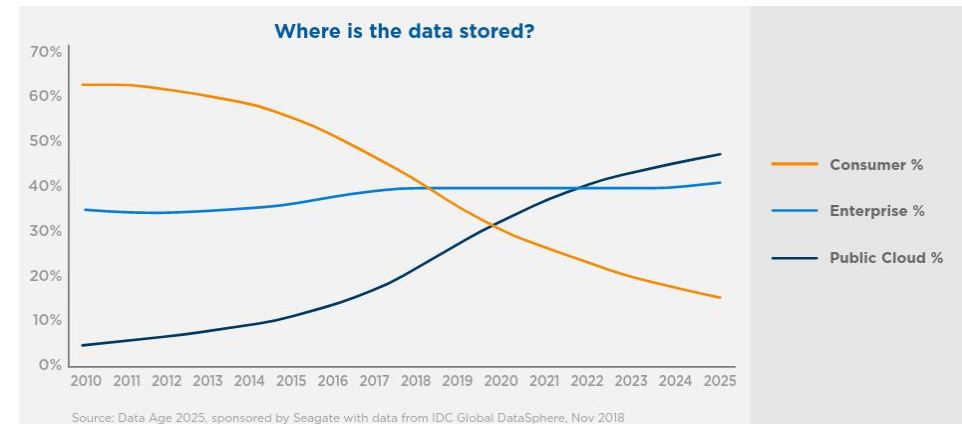
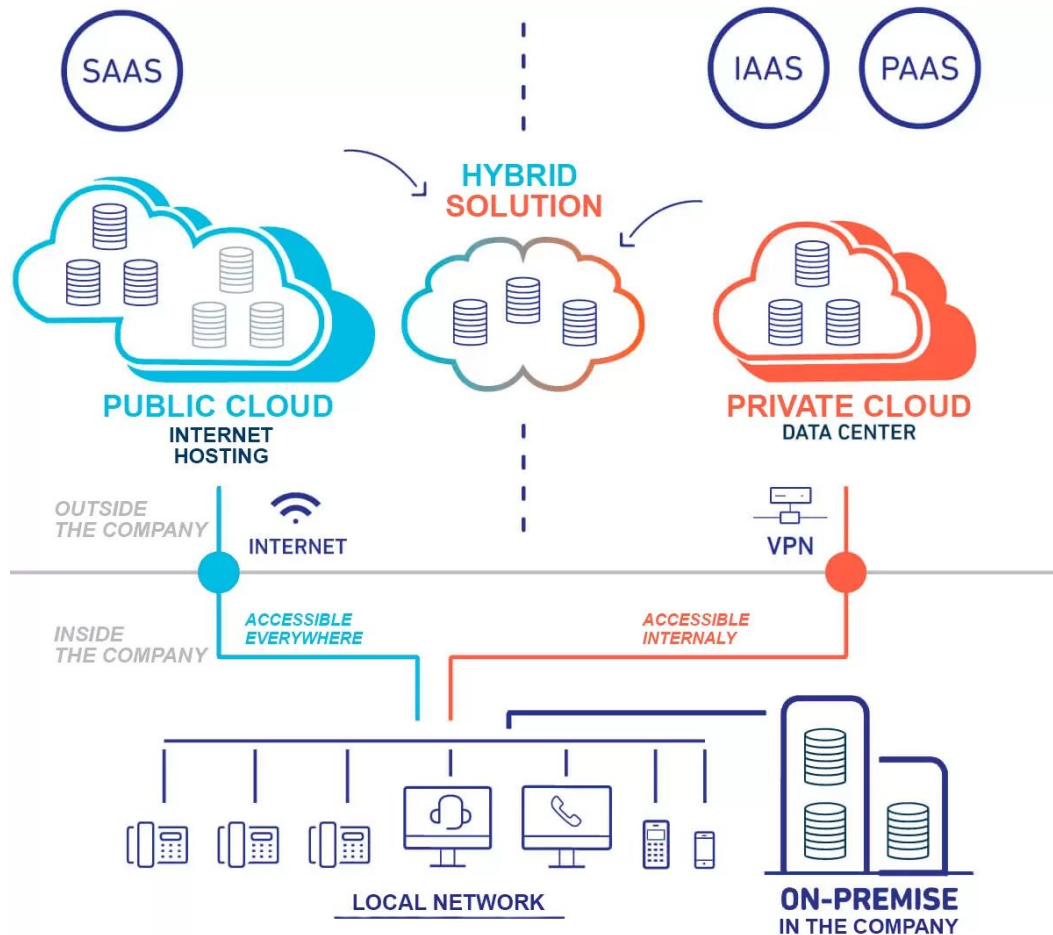
Wide Column Stores — Cassandra and HBase

Document databases — MongoDB

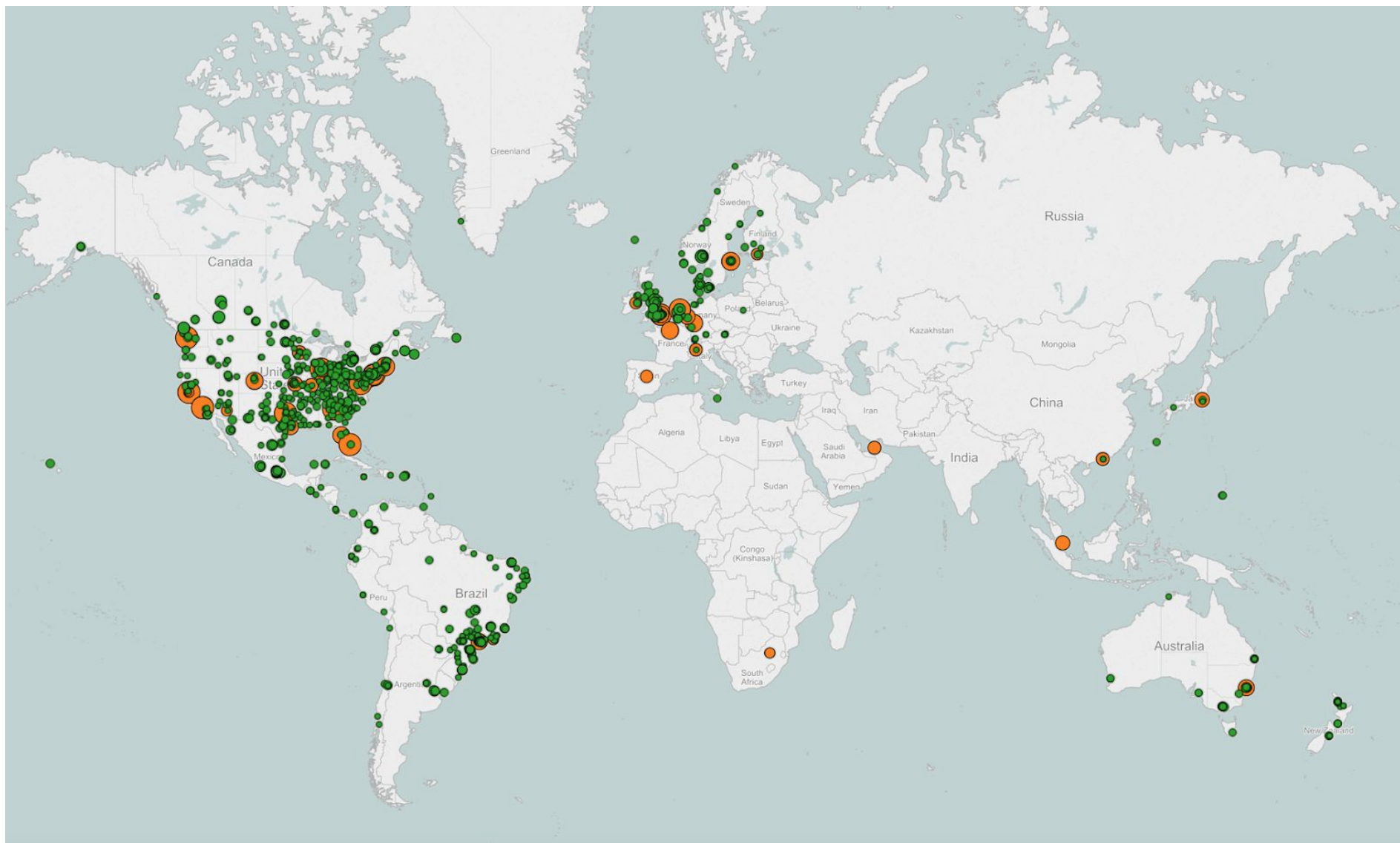
Graph databases — Neo4J and HyperGraphDB



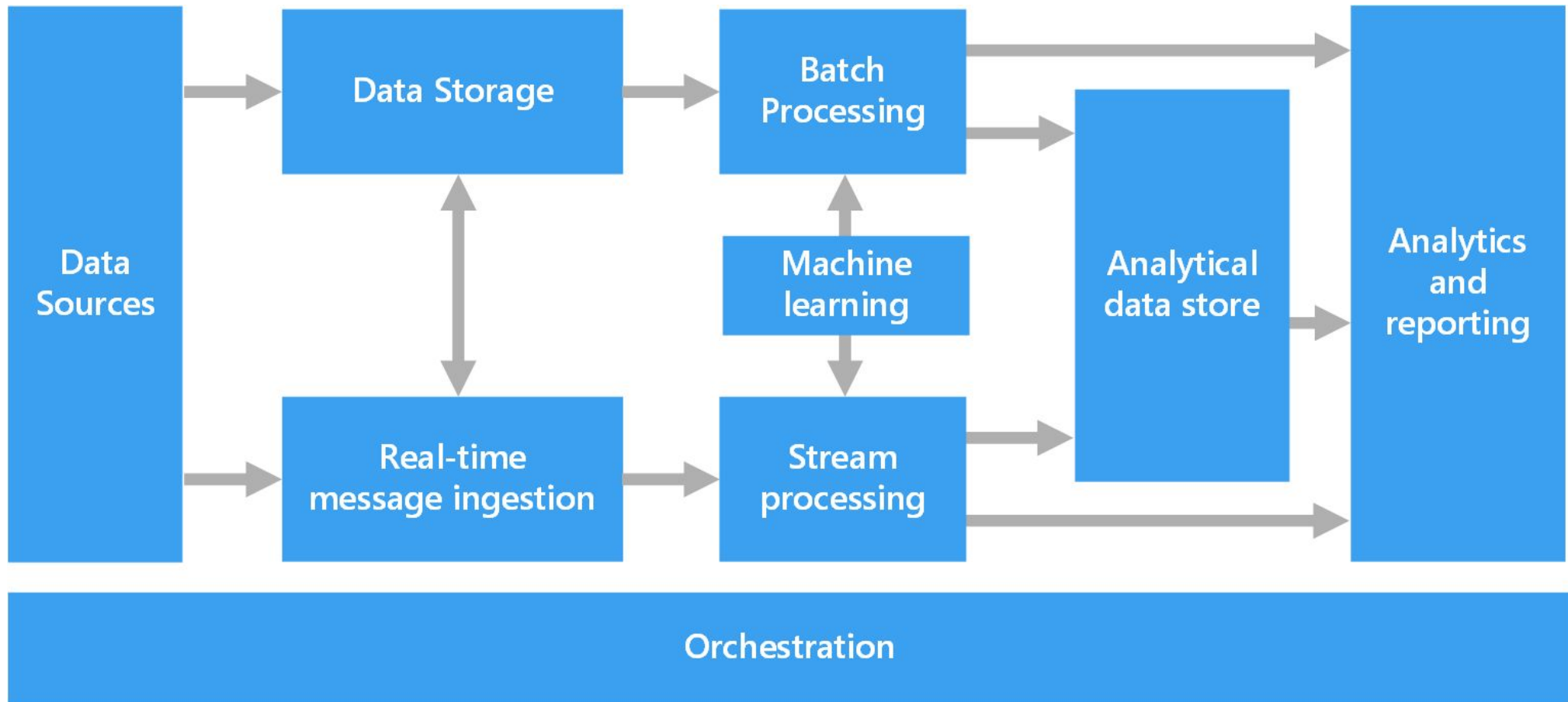
Cloud computing dominance



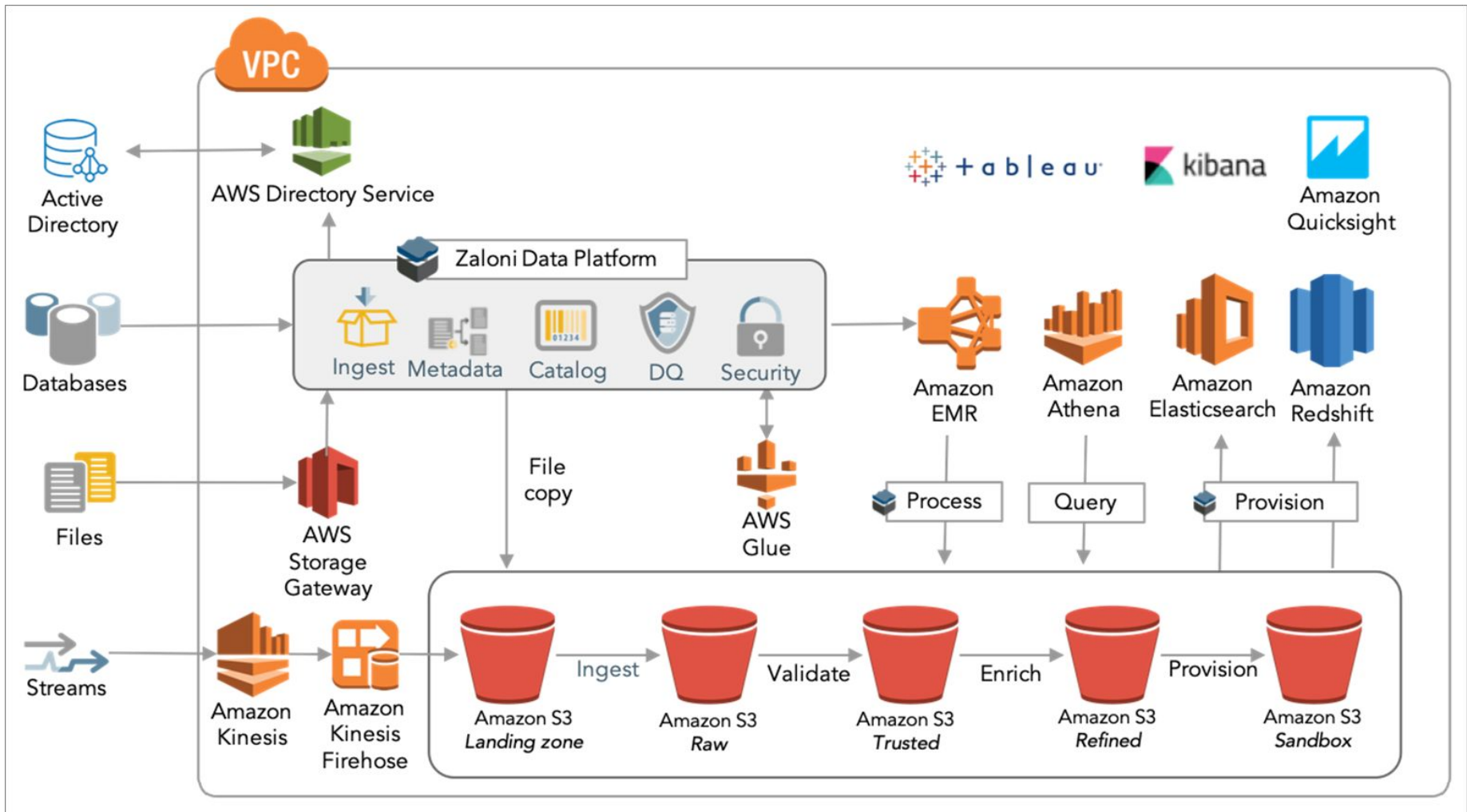
Distributed data: Netflix CDN



Modern data pipelines



Data pipelines in the cloud



ETL for big data with Hadoop and Spark

- ETL tools that can handle massive amounts of data
- data from distributed systems
- Hadoop
 - distributed storage
- Apache Spark
 - in-memory processing
- allow ETL pipelines to scale horizontally
 - distributes data and the computational workload across many nodes



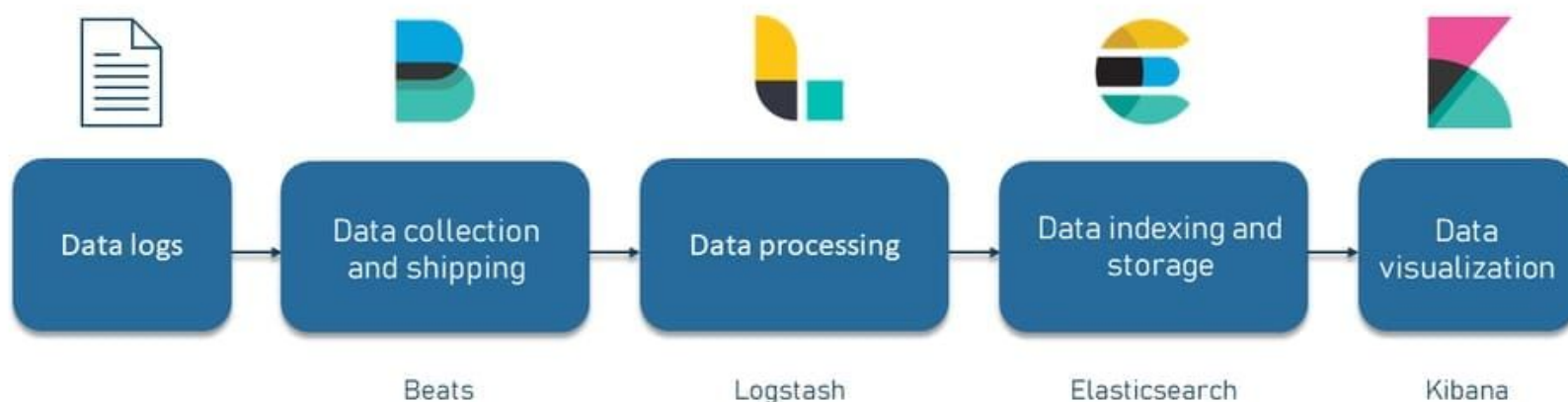
Real-time ETL with Kafka and Spark Streaming

- Traditional ETL pipelines often work in batch mode
- Real-time ETL is becoming increasingly important
- real-time ETL pipelines: data is extracted, transformed, and loaded continuously as new data flows in
- Apache Kafka
 - is a distributed streaming platform that enables real-time data ingestion
- Apache Spark Streaming
 - allows for real-time data processing
- use cases
 - monitoring financial transactions
 - real-time analytics
 - IoT applications



Loading with Elasticsearch

- a distributed search and analytics engine
- indexing structured and unstructured data for real-time analytics
- Use Cases: Real-time log analysis, full-text search, monitoring and alerting.
- ETL Role: "Load" destination in ETL workflows
- Advantages: Horizontal scalability, distributed architecture, and near-instant search results on large datasets.



Project: Design a data pipeline

- You are the 4 person data engineering team of a startup
- 2 - 4 page report on ETL/ELT pipeline choices
- Motivate and explain choices:
 - E: where are the data coming from?
 - T: how are the data being transformed?
 - L: how are the data loaded, stored, and used?
- Demo of example database
 - PostgreSQL, Mongo, other, your choice
 - Documented scripts to load and manipulate example data
 - Data should not be exhaustive, should demonstrate ETL choices

Report rigor	6
Report clarity	6
Demo data	4
Demo manipulation	4

Deadline:

Oct 11, 2024, end of day

Designing a data pipeline: extraction

- What are the various data sources?
- What is the format of the data from each source (e.g., CSV, JSON, XML, database tables)?
- Is the data streaming or can it be loaded in batches?
- What will the data look like on extraction?
- How do you verify the data's accuracy and completeness at the source?
- Are there any data access limitations or security constraints?
- How frequently is the data updated or changed at the source?
- Will you need to deal with incremental data extraction or full data loads?
- What are the volume and velocity of the data (e.g., terabytes per day, real-time streams)?

Designing a data pipeline: transformation

- What data cleansing steps are needed?
- Are there any business rules that need to be applied during transformation?
- Do you need to join or merge data from multiple sources?
- Are there any specific data formats or types that need to be converted?
- How will you handle any data inconsistencies or errors?
- Are there any dependencies between the transformation steps (e.g., one transformation requires another to be completed first)?
- Do you need to enrich the data by adding additional calculated fields?
- How will you track the changes to the data for auditing purposes?
- Will transformation happen before or after loading?

Designing a data pipeline: loading

- What is the target system for the data (e.g., data warehouse, data lake, database)?
- How often will the data be loaded (e.g., real-time, hourly, daily)?
- Should the data be appended to the existing dataset or replace it entirely?
- Are there any schema or structural requirements for the target system?
- How do you ensure data consistency and integrity during loading?
- What are the performance constraints?
- How will you handle schema changes in the target system?
- Is there a need for historical data tracking or versioning in the target?
- How will you monitor the loading process to ensure it runs successfully?

ETL or ELT

Data Format:

- ETL is needed for unstructured data to transform it into a relational format.
- ELT can be used if the data is already in a relational or flat format.

Data Size:

- ELT is suited for large datasets due to enhanced processing capabilities.
- ETL is typically used for smaller datasets.

Cost:

- ETL can be expensive as it involves physically moving data for processing.
- ELT can be more cost-effective, especially when leveraging cloud-based parallel processing without moving data.

Data Source:

- ELT is better for large, batch-wise data from cloud sources.
- ETL is preferred for streaming or messaging data.

Data Destination:

- ETL is often used when the source and destination systems differ.
- ELT is suitable when transferring data between identical systems or products.

Transformation Intensity:

- ELT is preferred for simple transformations and scalability.
- ETL is chosen for more complex transformations that need to be handled in smaller steps.

Transformation Location:

- In ETL, transformations occur before the data reaches the warehouse.
- In ELT, transformations are performed after data import, requiring more processing power but offering faster insights.

Almanapp: Long-Term Weather Forecast

- Almanapp provides users with weather and climate forecasts for specific regions over long periods (e.g., weeks or months). The app combines short-term weather forecasts and long-term climate predictions to visualize average weather conditions based on user preferences or location.
- Collect data from multiple sources like OpenWeatherMap, NOAA Climate Data, or other open APIs that provide both short-term and long-term weather forecasts.
- Identify how to aggregate weather data into useful insights.
- Create a workflow for real-time data extraction and periodic long-term climate updates.



Veloù: Where Your City Bikes Are

- Veloù tracks real-time bike availability in a city, correlates it with OpenStreetMap data to visualize common bike paths, and provides statistics on when bike stations are often full or empty.
- Collect real-time bike usage data from city bike-sharing platforms (many cities provide this data openly) and geographical data from OpenStreetMap.
- Build a pipeline that processes both real-time bike station data and static map data.
- Correlate bike usage with common bike paths and times of day.
- Ensure high availability for real-time data tracking.



Airlife: Discover an Airplane's History

- Airlife allows users to track the lifetime history of a specific airplane, including total distance flown, estimated carbon footprint, and its most recent location.
- Pull data from OpenFlights (flight routes, airports) and live flight tracking APIs (e.g., FlightAware, OpenSky Network).
- Load data into a database to allow quick queries about an airplane's history. Visualize key statistics on a dashboard.
- Estimate carbon footprint using factors like aircraft type and distance flown.
- Handle frequent updates to reflect current location and flights.



EcoTrack: Urban Environmental Monitoring

- Track and visualize environmental metrics such as air quality, noise pollution, and temperature in urban areas.
- Gather real-time environmental data (e.g., air quality index, noise levels) and correlate it with weather conditions. Open air quality APIs (e.g., AirVisual), environmental sensors, public transportation schedules, weather data.
- Store the data in a time-series database for real-time analytics.
- Visualize environmental quality in different areas of the city.



Foodwatch: Farm-to-Table Tracker

- Track the journey of food from farms to retail stores, providing transparency in the supply chain, including origin, transportation, and environmental impact.
- Pull data on food production from open agricultural databases and logistics information from shipping APIs. Agricultural data from public sources, logistics data from open platforms, weather data for transportation routes. Global Open Data for Agriculture and Nutrition (GODAN), European Data Portal.
- Load into a system that tracks the movement of each batch of food, making it queryable by store or product. Visualize product history and calculate distances and carbon cost.



Project: Design a data pipeline

- You are the 4 person data engineering team of a startup
- 2 - 4 page report on ETL/ELT pipeline choices
- Motivate and explain choices:
 - E: where are the data coming from?
 - T: how are the data being transformed?
 - L: how are the data loaded, stored, and used?
- Demo of example database
 - PostgreSQL, Mongo, other, your choice
 - Documented scripts to load and manipulate example data
 - Data should not be exhaustive, should demonstrate ETL choices

Report rigor	6
Report clarity	6
Demo data	4
Demo manipulation	4

Deadline:

Oct 11, 2024, end of day