# (No)SQL

Databases

# Agenda

- Who am I
- Classification of Databases
- Types of NoSQL
- Under the hood (Distributed systems)
- Bechmarking Tools
- Personal Journey
- Discussions\Q&A

# Who am I

- Suyambu Ganesh Thanga Nadar
    - Call sign : **Ganesh**
- Married, two children, living in Amstelveen
- Senior Java Dev/Tech Lead/Solution Architect (Cloud\µServices\IoT)
- Joined Quby in 2016\Merged with Eneco in 2021
- Tech enthusiast\Innovative\fascinated by BigData and IoT
- Prof. career spanning 15+ years

# Classification

- SQL - Relational Databases
- NoSQL -  Not Only SQL
- NewSQL - NoSQL + SQL

# SQL

- Relational Databases
- Based on Relational Algebra not because of Foreign key
- Predefined Schema in Tabular
- ACID Properties
  - Atomicity (Commits entire operation or rollbacks to prev. state)
  - Consistency (Maintains the data integrity, same view for all clients)
  - Isolation (R/W operations are perfomed in isolation)
  - Durability (Successfull commits should be permanent)
- Vertical scalability
- SQL as the query language
- OLTP (Transaction Processing - Tables)
- OLAP (Business Intelligence - Cubes)
- Synchronous Inserts and Updates
- Storage + Query + Index engines






# NoSQL






- Non – relational or Distributed databases
- Dynamic schema
- Document\Key Value\Graph\Wide Column store
- Unstructured query language
- Based on CAP theorem
  - Consistency (Same view for all clients)
  - Availability (Highly available even half of machine is down)
  - Partition tolerant (The system works even when there is partial network failure)
- Not suited for huge Transactional workloads
- Horizontally scalable (115k nodes and over 10PB of data)
- Web 2.0 (user generated content and web)
- Asynchronous inserts and updates

Cloud
Bigtable

amazon
DynamoDB

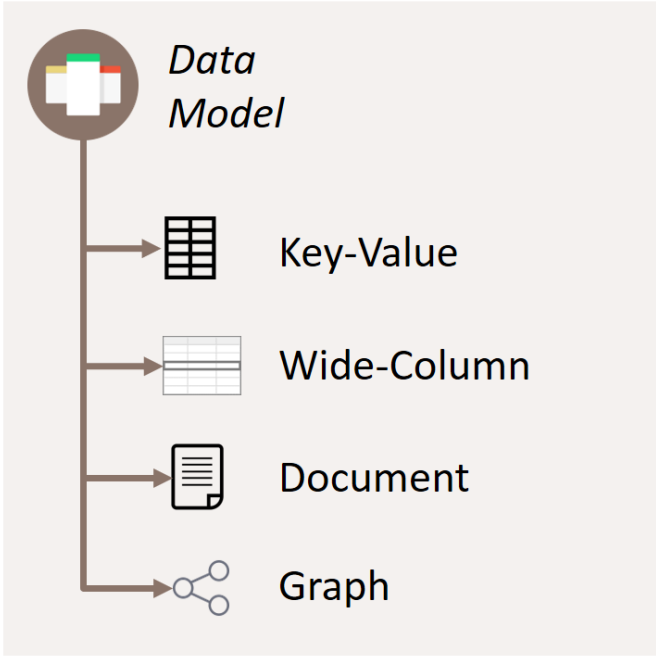
# NewSQL

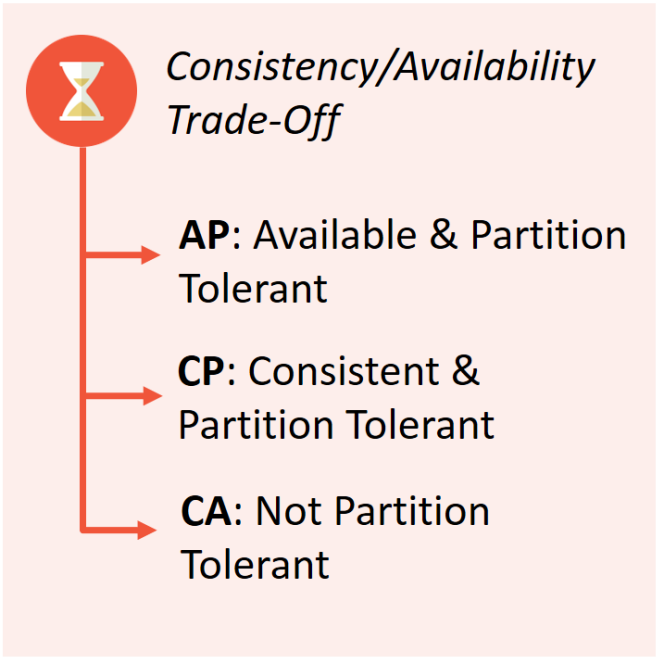
- New Generation of SQL databases
- ACID properties + Distributed Goodness
- Compromise on latency
- SQL and Table structure
- Horizontally scalable
- Globally distributed




# Type of NoSQL

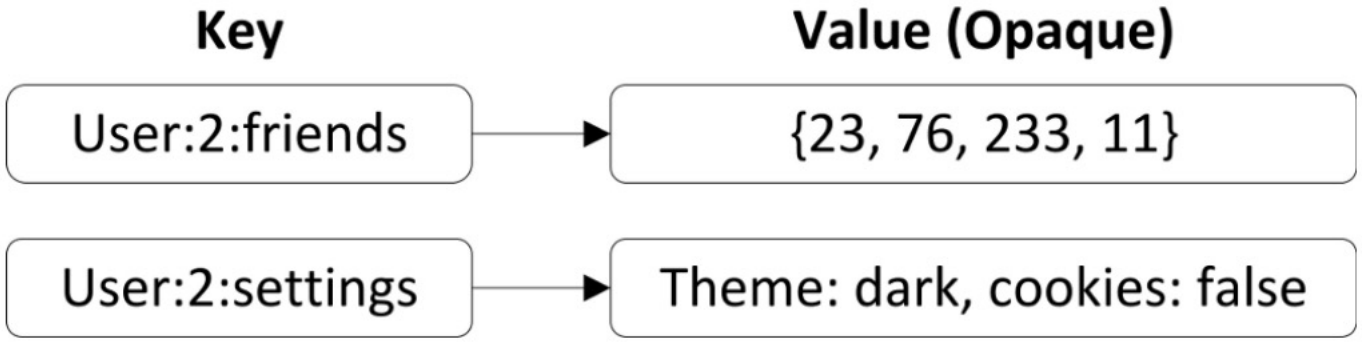
- Two approaches
  1. DataModel
  2. CAP theorem (Professor Eric A. Brewer)



# Key value Store

- KV pairs
- Unique Keys
- Schemaless
- Fast read and writes
- Inmemory storage
- Caching
- Counters
- FDB (billions of databases)



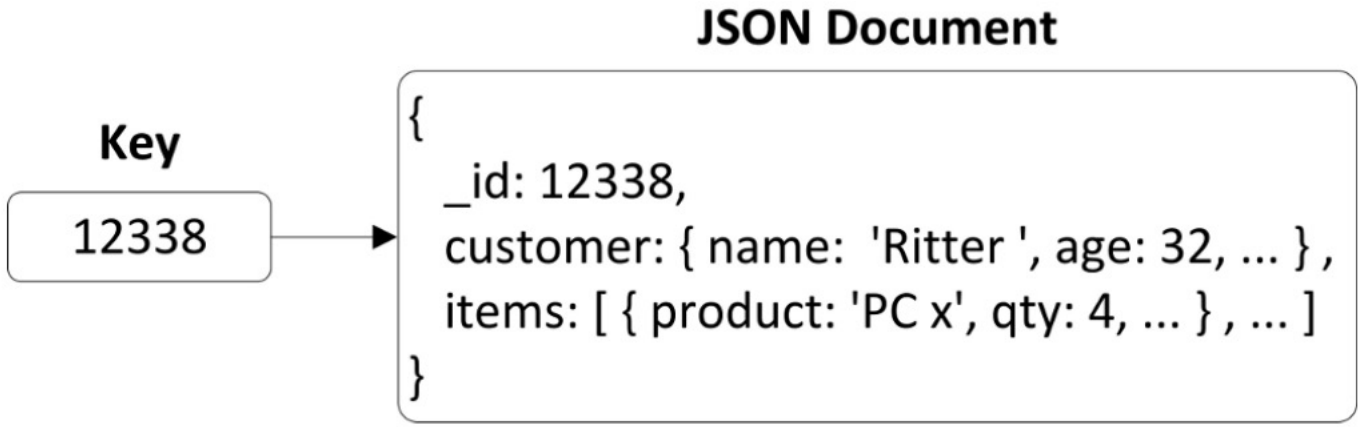

redis

MEMCACHED

FOUNDATIONDB

# Document Store

- Key Value
- Semi-structured JSON
- Full document or part of Do
- Aggregation
- Full text search
- Analytics
- Product catalog
- Product search




mongoDB

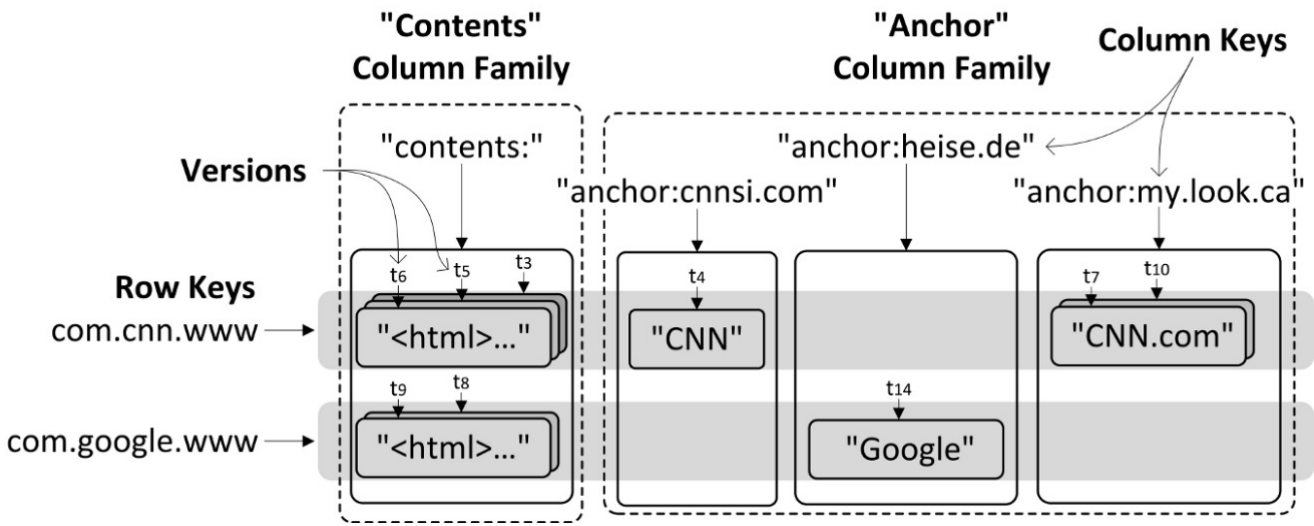
Couchbase

elasticsearch

CouchDB
relax

# Wide column store

- Similar to Map structure
- Row keys and Column keys
- Column families
- Billions of columns and rows
- Petabyte scale
- 100 millions of write/sec
- Google Bigtable pioneered
- User generated content
- Events\Time series data



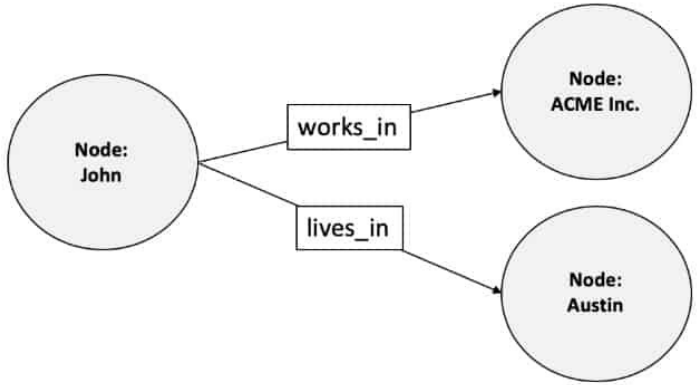

APACHE HBASE

cassandra

accumulo™

# Graph store

- Graph theory
- Nodes, edges and properties
- Efficient join
- Fast traversal
- Millions of connections/second
- Recommendation Engine
- Master data management
- Fraud detection
- Identity and Access management



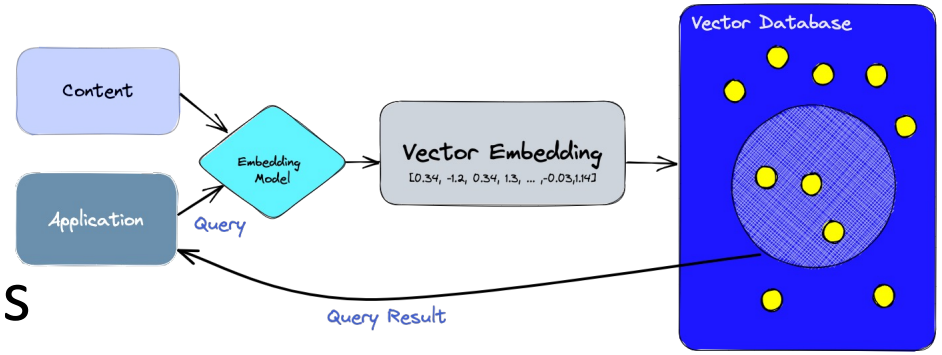

neo4j

JanusGraph

Dgraph
The GraphQL Database

# Vector Databases (HOT Topic right now)

- Stores vector embeddings
- Embeddings are encoded representation of text, images, audio
- Stored as a array of numbers with the semantics
- Vector db supports similarity search
- Used in AI applications like ChatGPT
- Highly scalable, low latency, Real time updates

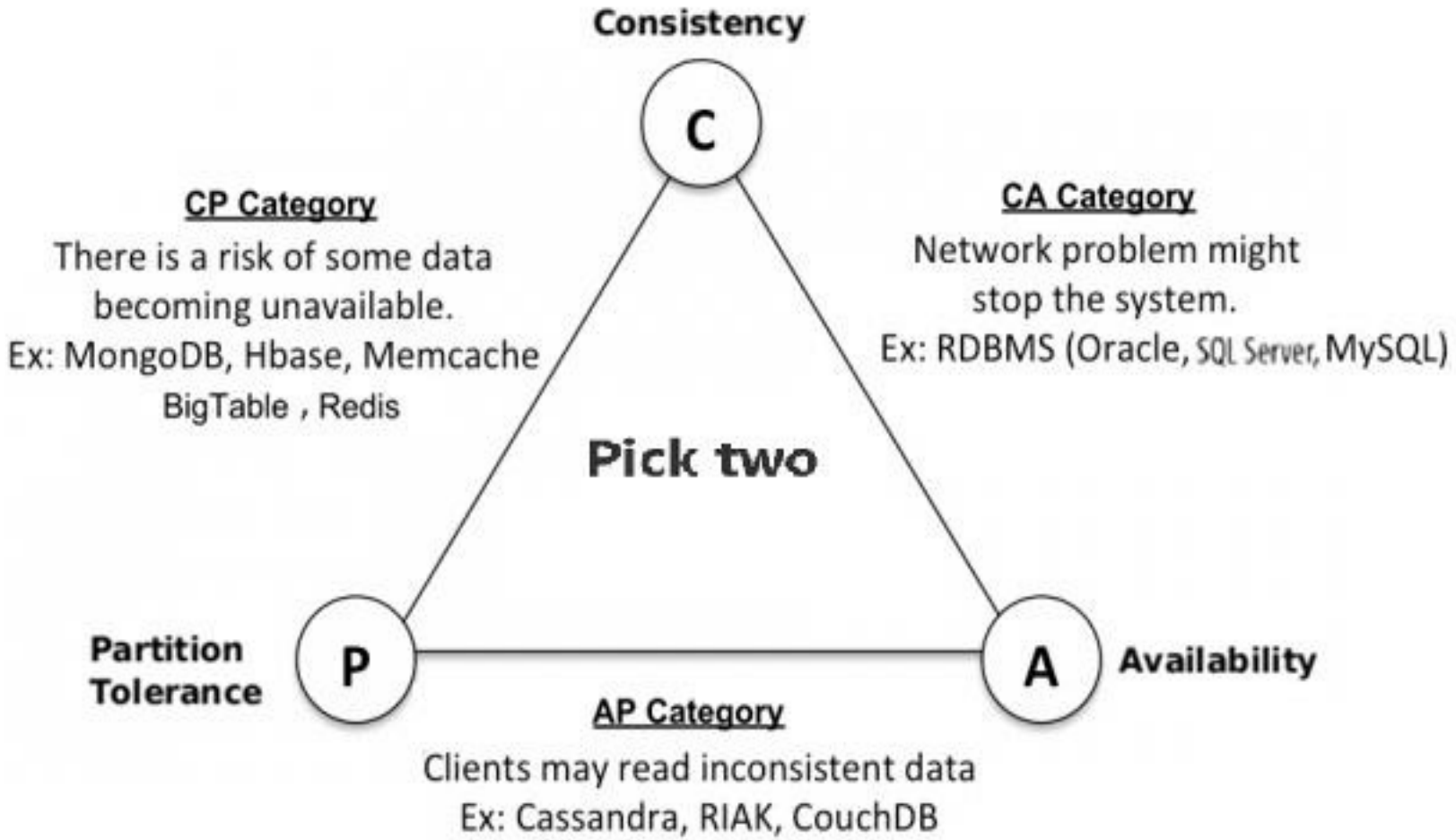



vespa Pinecone Weaviate cassandra

# CAP Theorem

- Defines upper bound guarantees on Distributed systems
  - **Consistency (C)**:
    - Reads and writes are always executed atomically and are strictly consistent.
    - All clients have the same view on the data at all times.
  - **Availability (A)**:
    - Every non-failing node in the system can always accept read and write requests by clients and will eventually return with a success response.
  - **Partition-tolerance (P)**:
    - The system upholds the previously displayed consistency guarantees and availability in the presence of message loss between the nodes or partial system failure.

# CAP Theorem


Consistency
C
CP Category
There is a risk of some data becoming unavailable.
Ex: MongoDB, Hbase, Memcache
BigTable , Redis
CA Category
Network problem might stop the system.
Ex: RDBMS (Oracle, SQL Server, MySQL)
Pick two
Partition Tolerance
P
A
Availability
AP Category
Clients may read inconsistent data
Ex: Cassandra, RIAK, CouchDB

# Comparision

| | Funct. Req. | | | | | | | | Non-Funct. Req. | | | | | | | | | | Techniques | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Scan Queries | ACID Transactions | Conditional Writes | Joins | Sorting | Filter Queries | Full-Text Search | Analytics | Data Scalability | Write Scalability | Read Scalability | Elasticity | Consistency | Write Latency | Read Latency | Write Throughput | Read Availability | Write Availability | Durability | Range-Sharding | Hash-Sharding | Entity-Group Sharding | Consistent Hashing | Shared-Disk | Transaction Protocol | Sync. Replication | Async. Replication | Primary Copy | Update Anywhere | Logging | Update-in-Place | Caching | In-Memory Storage | Append-Only Storage | Global Indexing | Local Indexing | Query Planning | Analytics Framework | Materialized Views |
| MongoDB | x | | x | | x | x | x | x | x | x | x | | x | x | x | x | x | | x | x | x | | | | | x | x | | x | | | x | x | | | x | x | x | |
| Redis | x | x | x | | | | | | | | x | | x | x | x | x | x | | x | | | | | | | | x | x | | x | | x | | | | | | | |
| HBase | x | | x | | x | | | x | x | x | x | x | x | x | | x | | | x | x | | | | | | x | | x | | x | | x | | x | | | | | |
| Riak | | | | | | | x | x | x | x | x | x | | x | x | x | x | x | x | | x | | x | | | | x | | x | x | x | x | | | x | x | | x | |
| Cassandra | x | | x | | x | | x | x | x | x | x | x | | x | | x | x | x | x | | x | | x | | | | x | | x | x | | x | | x | x | x | | x |
| MySQL | x | x | x | x | x | x | x | x | | | x | | x | | | | | | x | | | | | x | x | | x | x | | x | x | x | | | | x | x | | |

# Under the HOOD of NoSQL

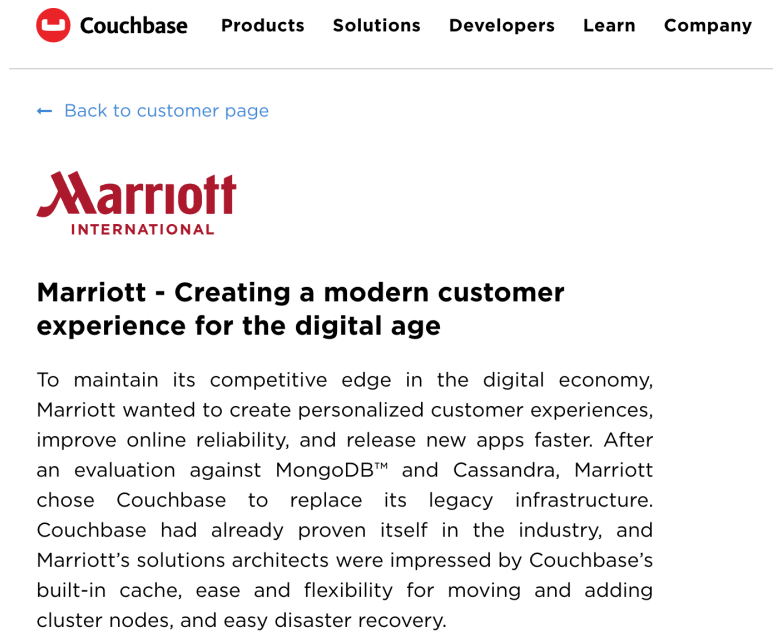
- Distributed computing
  - Responsibilities shared across many computers in a network
- Shared nothing architecture
  - Each server instance manages its own data
- Consensus Algorithm
  - leader election
  - data sharding\ data partitioning
  - Metadata sharing
  - example algorithms
    - Paxos (cassandra)
    - Raft (Neo4j, YugabyteDB)
    - ZAB (Zookeeper)

# Benchmarking Tools

- YCSB : Yahoo cloud service benchmark
  - https://github.com/brianfrankcooper/YCSB
- TLA+
  - Is a spec to design systems and algorithms and verifying
  - https://learntla.com/introduction/
- TPC
  - Transaction Processing Performance Council
  - http://www.tpc.org/
- Jepsen
  - To improve the safety of the data.
  - Verify databases on ACID or CAP compliance
  - https://jepsen.io/

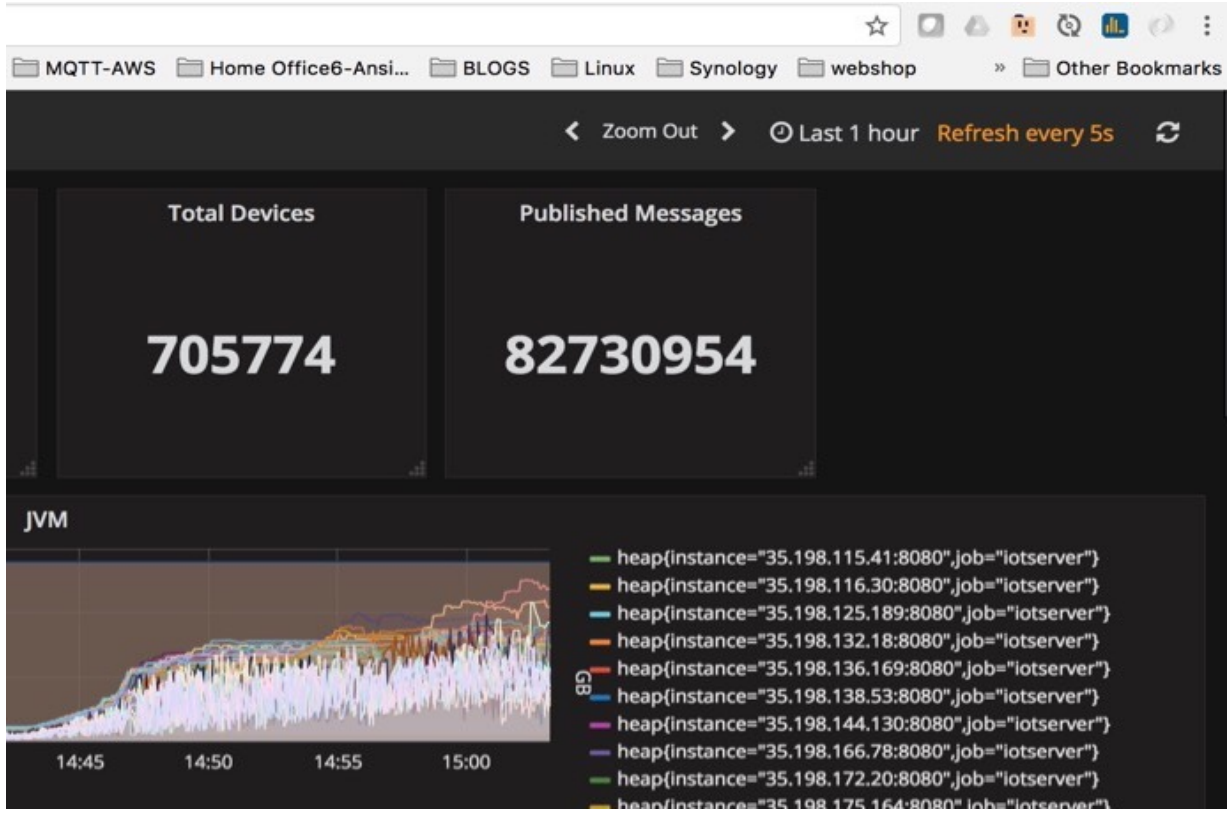
# Personal Journey with NoSQL databases

- Started in 2012
  - Designed logging and tracing system for KLM
  - Storing and Indexing 70 GB of XML data in 4 minutes
  - Used Cassandra 0.8 in 3 un-used desktop machines
  - 1 month data retention of 6 TB of compressed XML
  - Searching trace by PNR, TicketNumber in < 30ms
- In 2013
  - Evaluated MongoDB, Cassandra and CouchBase
  - Recommended Couchbase to Hotel Chain Marriott
  - Still couchbase is in use at Marriott



# Personal Journey with NoSQL databases

- Created MQTT Broker
- Cassandra as backend
- Simulated 700K MQTT clients
- 82 Million messages in 15 minutes
- 90K msgs/second



# Q\A