MAKING ENTERPRISE DATA AVAILABLE IN REAL TIME WITH ELASTICSEARCH

Yann Cluchey

~~CTO @ Cogenta~~

CTO @ GfK Online Pricing Intelligence

INTERNATIONAL SOFTWARE DEVELOPMENT CONFERENCE

What is Enterprise Data?

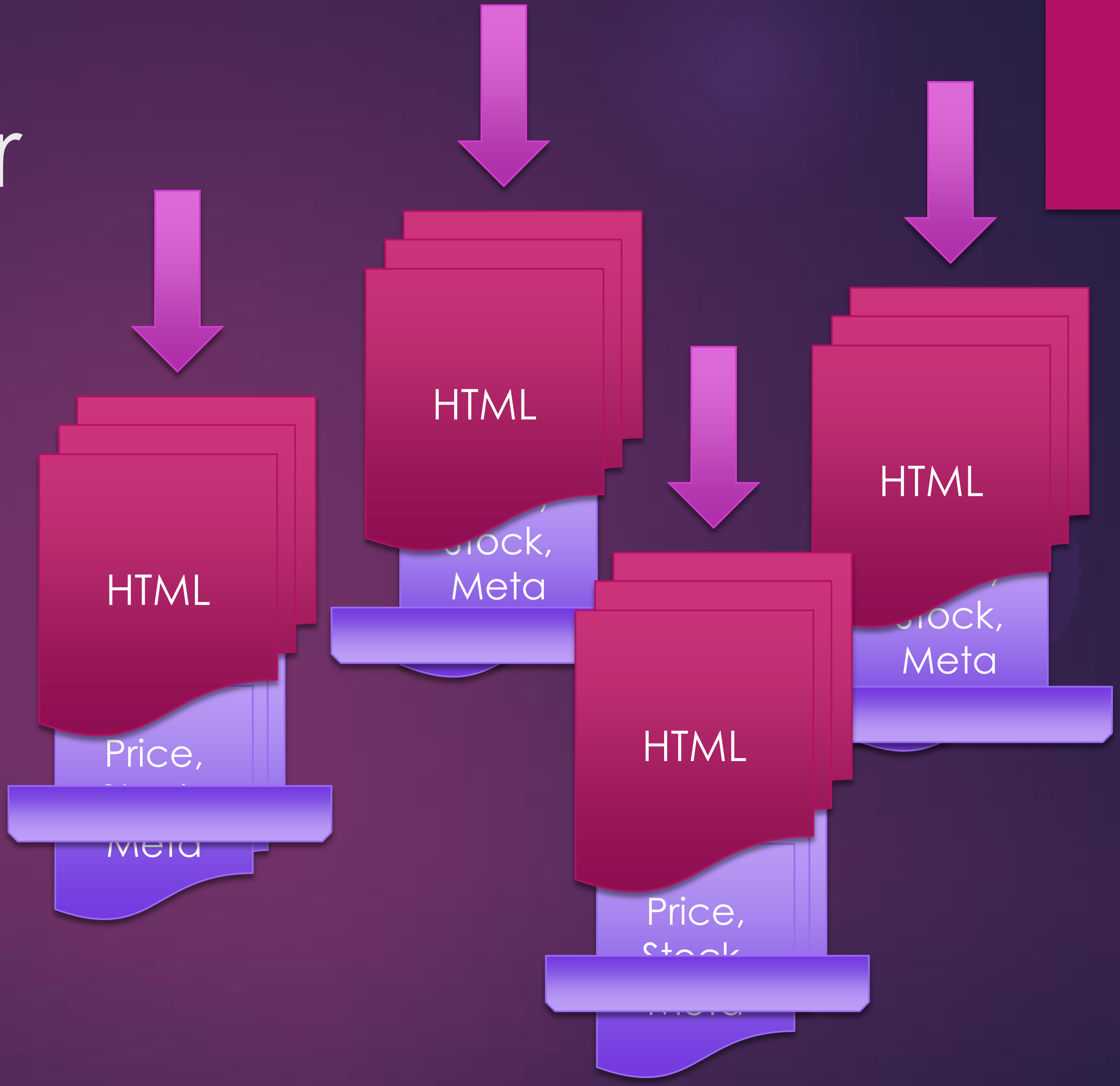
What is Enterprise Data?

Online Pricing Intelligence

1. Gather data from 500+ of eCommerce sites
2. Organise into high quality market view
3. Competitive intellige

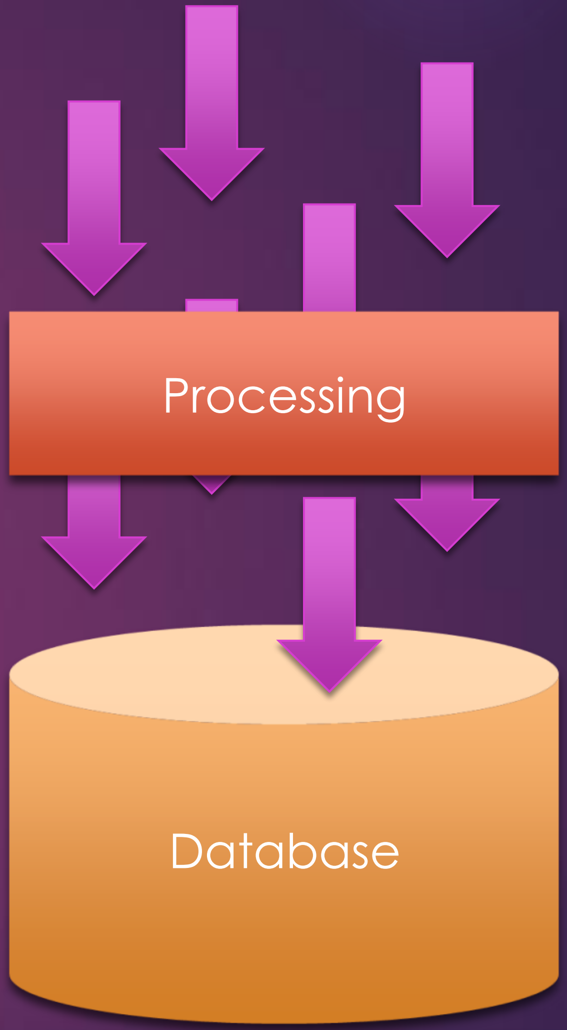
Custom Crawler

- Parse web content
- Discover product data
- Tracking 20m products
- Daily+



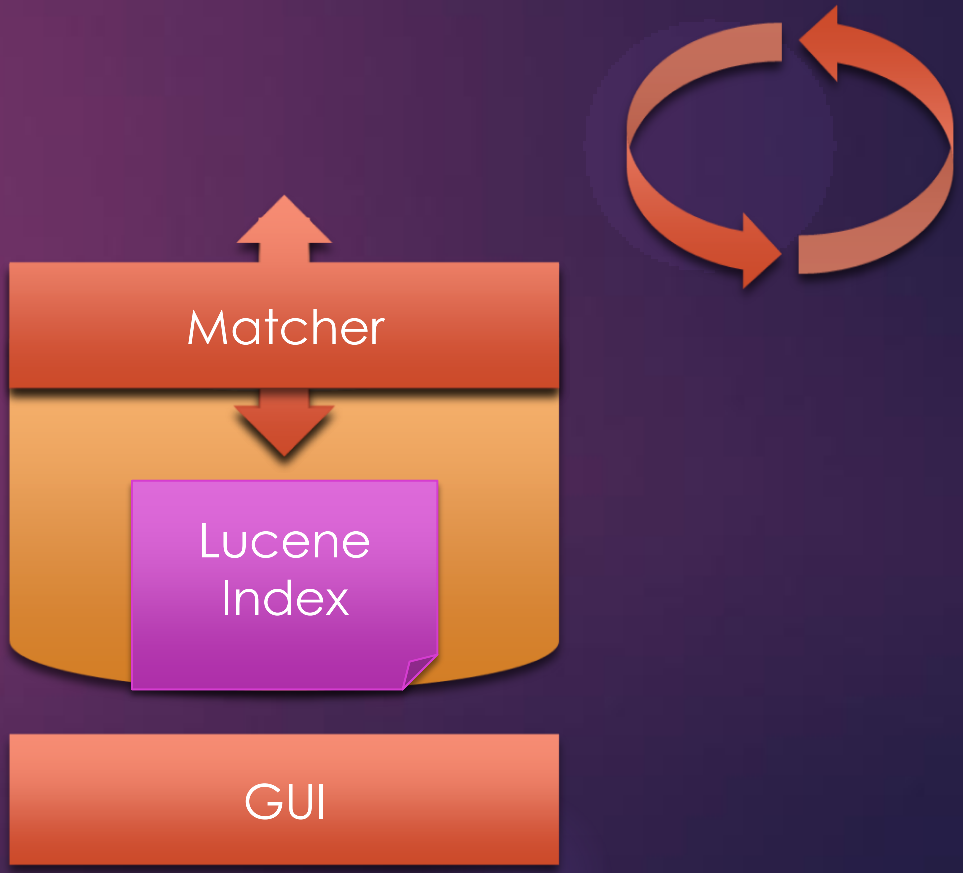
Processing, Storage

- Enrichment
- Persistent Storage
- Product Catalogue
- + time series data



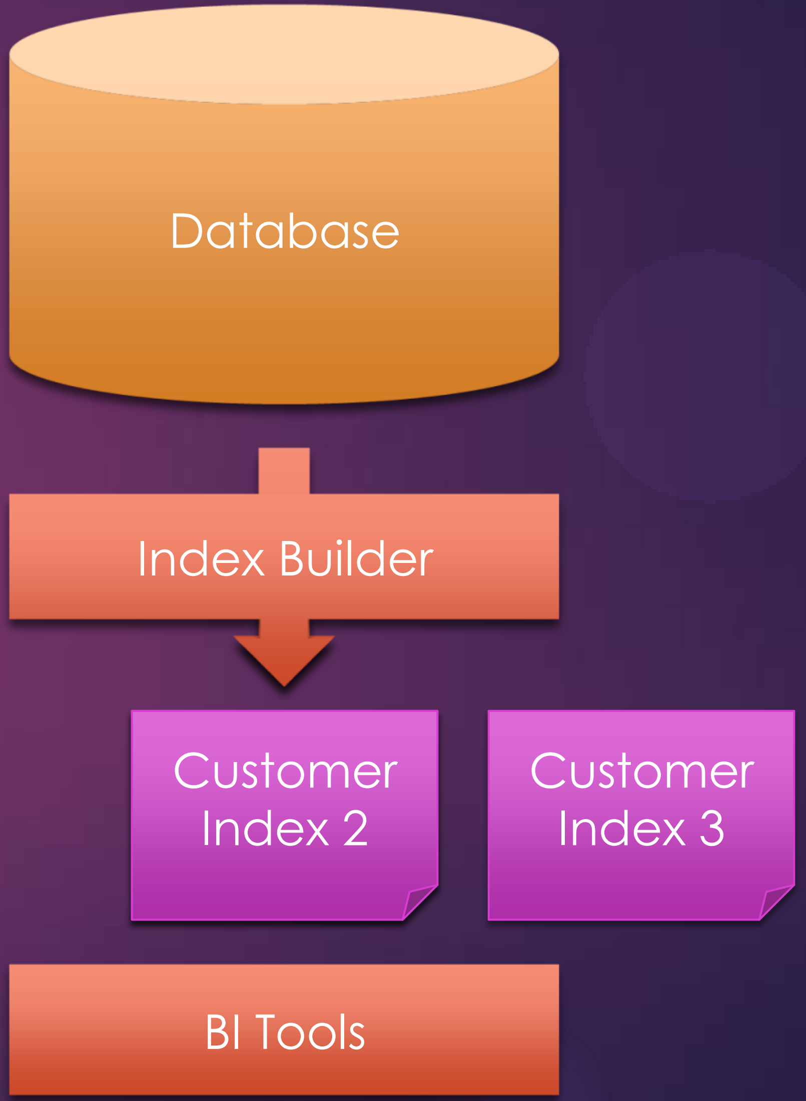
Thing #1 - Detection

- Identify distinct products
- Automated information retrieval
- Lucene + custom index builder
- Continuous process
- (Humans for QA)



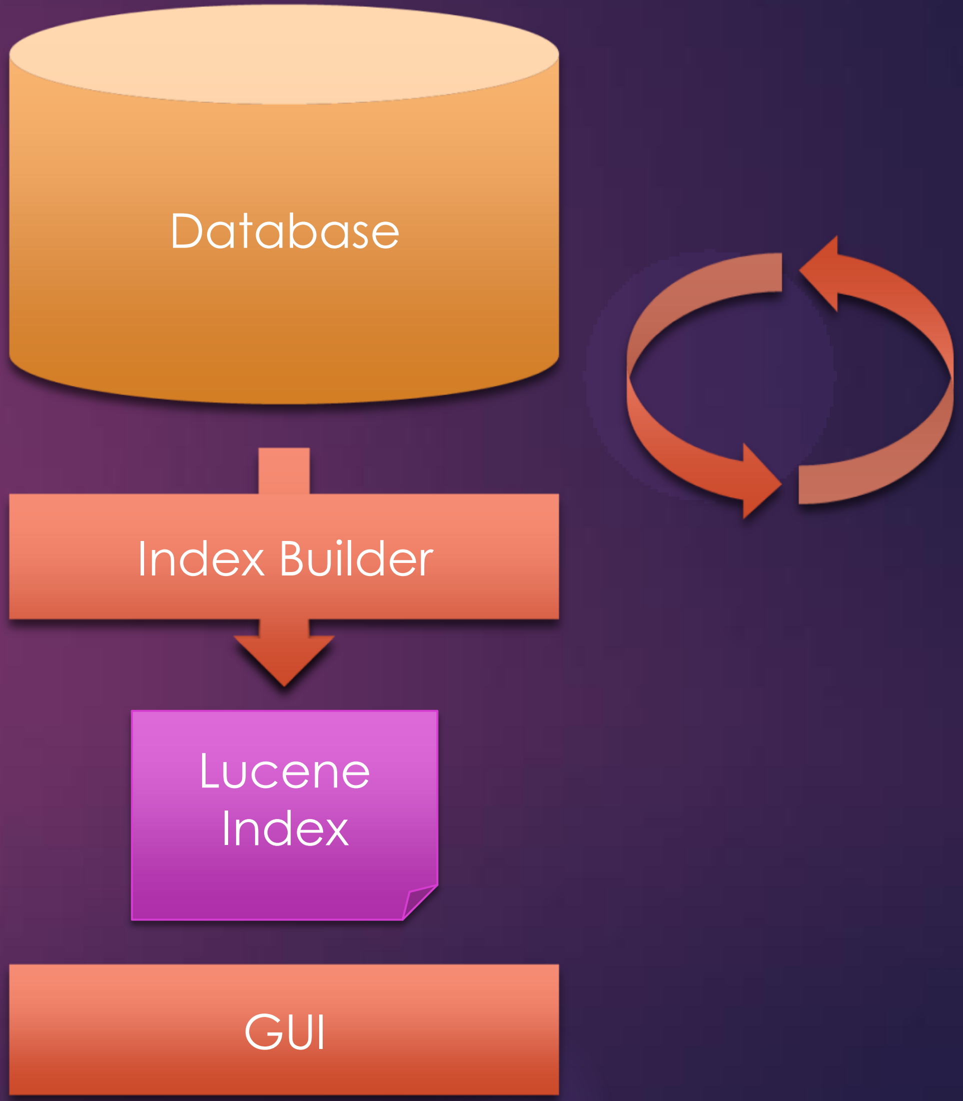
Thing #2 - BI Tools

- Web Applications
- Also based on Lucene
- Batch index build process
- Per-customer indexes



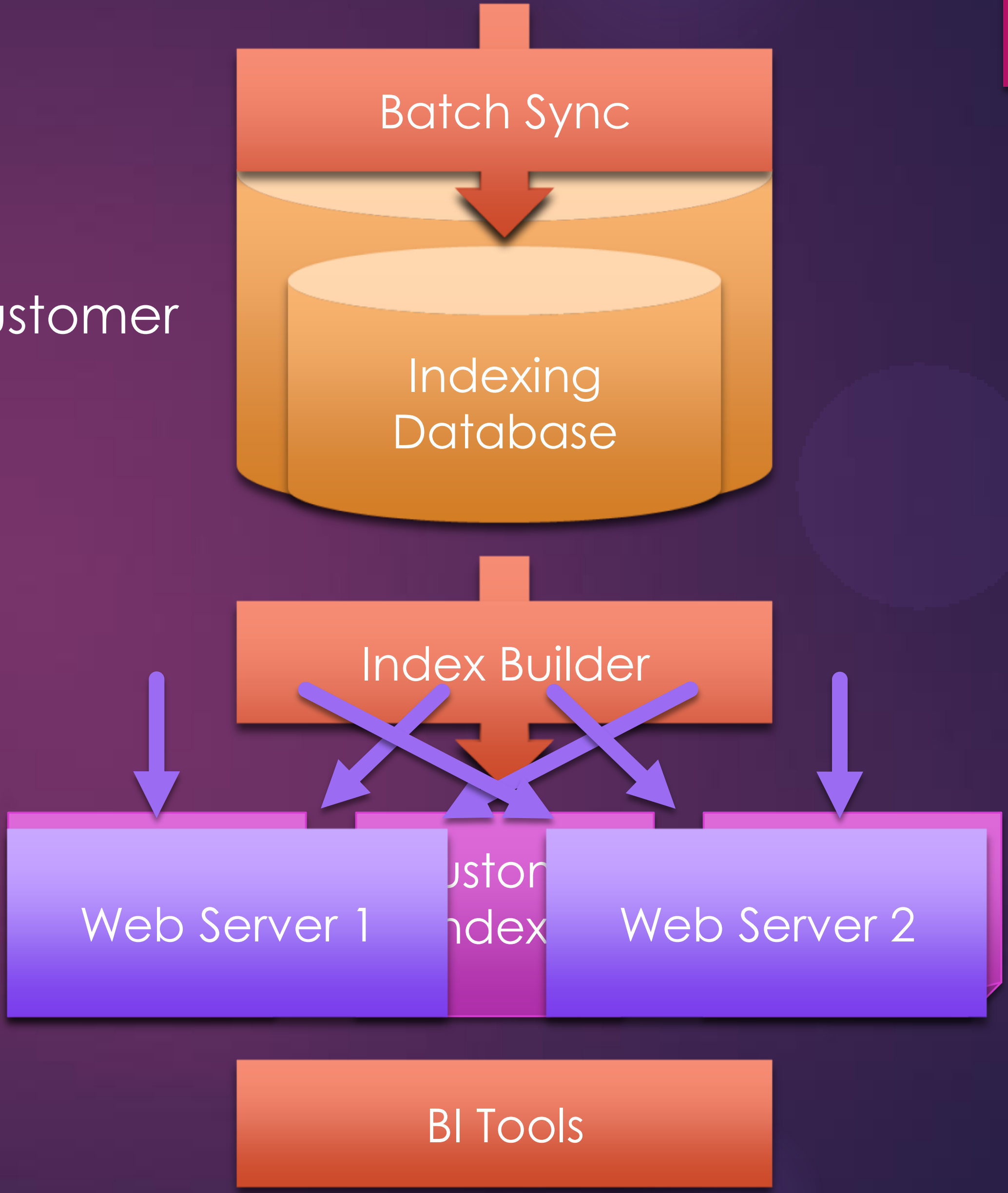
Thing #1 - Pain

- Continuously indexing
- Track changes, read back out to index
- Drain on performance
- Latency, coping with peaks
- Full rebuild for index schema change or inconsistencies
- Full rebuild doesn't scale well…
- Unnecessary work..?



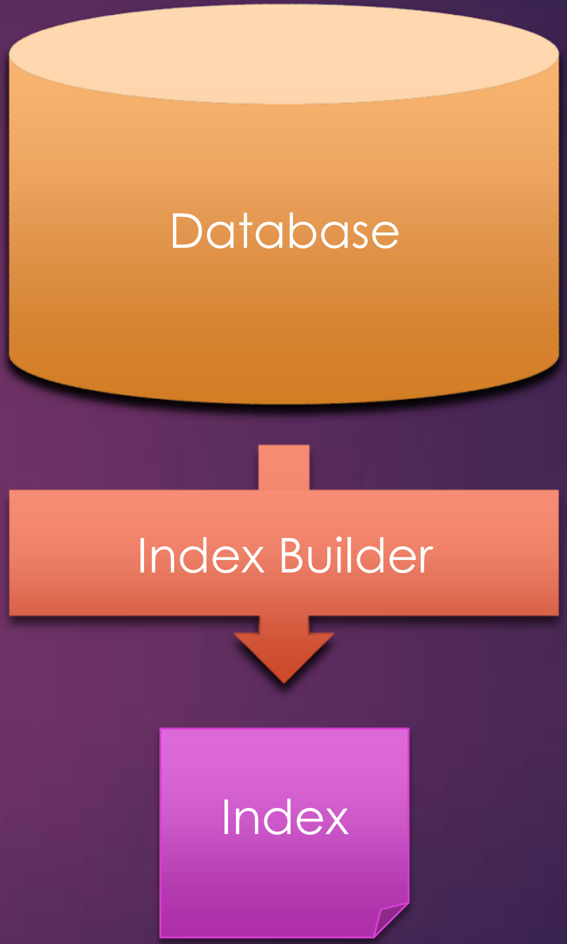
Thing #2 - Pain

- Twice daily batch rebuild, per customer
- Very slow
- Moar customers?
- Moar data?
- Moar often?
- Data set too complex, keeps changing
- Index shipping
- Moar web servers?



Pain Points

- As data, customers scale, processes slow down
- Adapting to change
- Easy to layer on, hard to make fundamental changes
- Read vs write concerns
- Database Maintenance



Goals

- Eliminate latencies
- Improve scalability
- Improve availability
- Something achievable

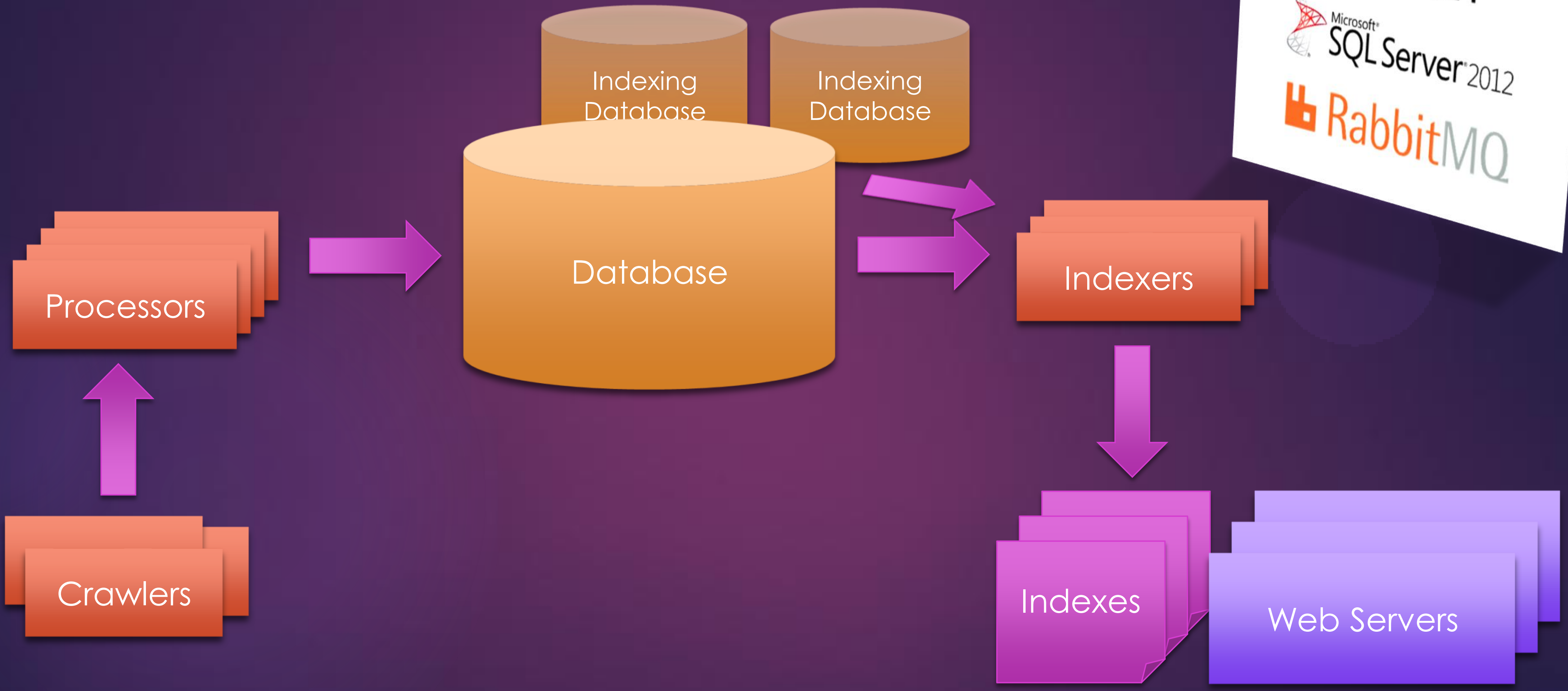
- Your mileage will vary

elasticsearch

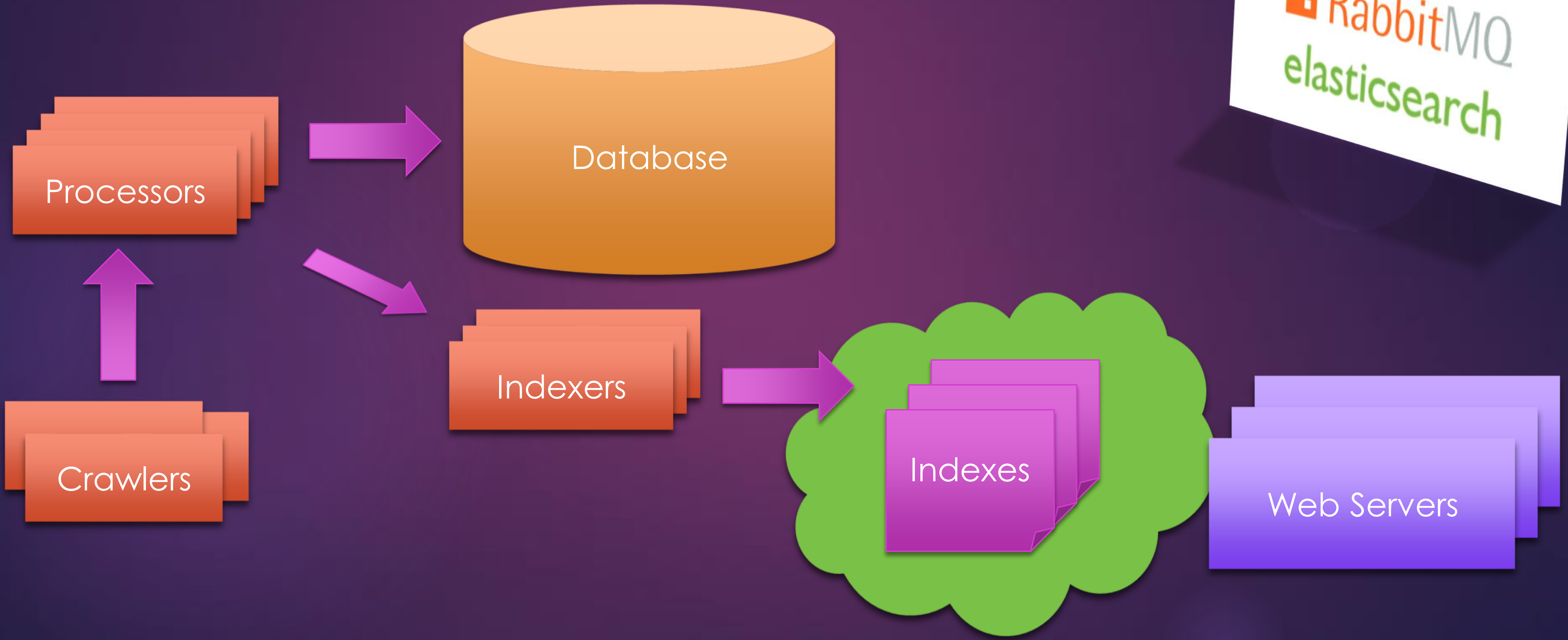
- Open source, distributed search engine
- Based on Lucene, fully featured API
- Querying, filtering, aggregation
- Text processing / IR
- Schema-free
- Yummy
  (real-time, sharding, highly available)

- Silver bullets not included

Our Pipeline



Our New Pipeline



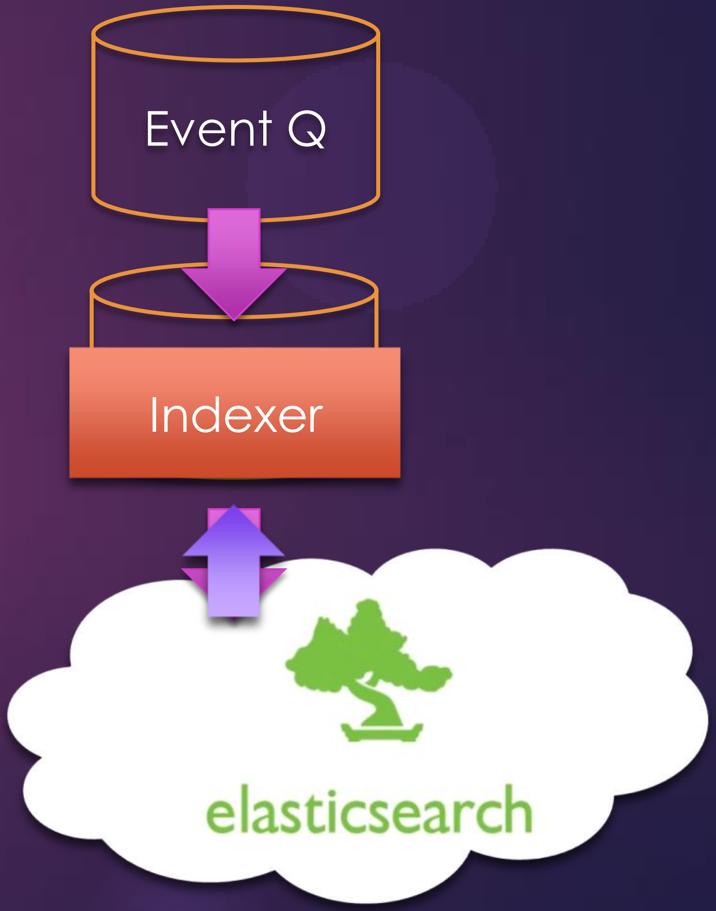
Microsoft .NET
Microsoft SQL Server 2012
RabbitMQ
elasticsearch

Event Hooks

- Messages fired OnCreate.. and OnUpdate
- Payload contains everything needed for indexing
  - The data
  - Keys (still mastered in SQL)
  - Versioning
- Sender has all the information already
- Use RabbitMQ to control event message flow
- Messages are durable

Indexing Strategy

- RESTful API (HTTP, Thrift, Memcache)
  - Use bulk methods
  - They support percolation
- Rivers (pull)
  - RabbitMQ River
  - JDBC River
  - Mongo/Couch/etc. River
- Logstash



Model Your Data

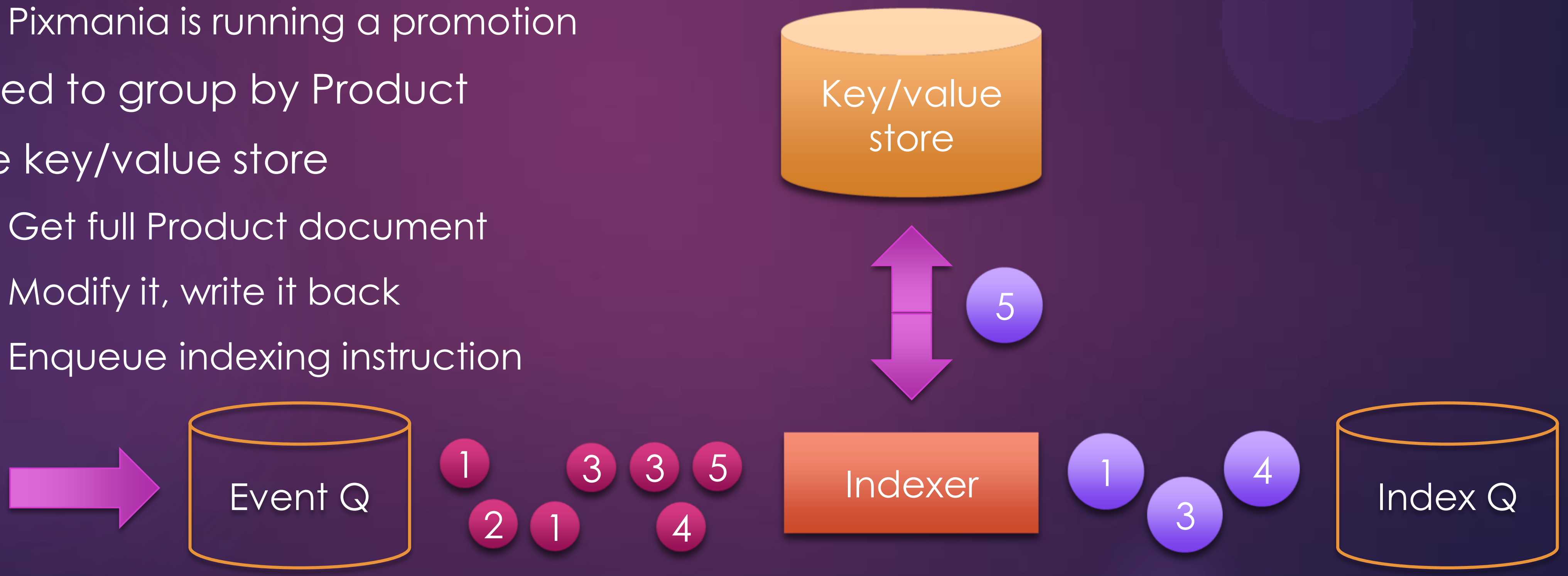
- What’s in your documents?
- Database = Index
  Table = Type ...?

- Start backwards
  - What do your applications need?
  - How will they need to query the data?

- Prototyping! Fail quickly!
- elasticsearch supports Nested objects, parent/child docs

Joins

- Events relate to line-items
  - Amazon decreased price
  - Pixmania is running a promotion
- Need to group by Product
- Use key/value store
  - Get full Product document
  - Modify it, write it back
  - Enqueue indexing instruction



Where to join?

- elasticsearch
  - Consider performance
  - Depends how data is structured/indexed (e.g. parent/child)
  - Compression, collisions
- In-memory cache (e.g. Memcache)
- Persistent storage (e.g. Cassandra or Mongo)
- Two awesome benefits
  - Quickly re-index if needed
  - Updates have access to the full Product data
- Serialisation is costly

Synchronisation & Concurrency

- Fault tolerance
  - Code to expect missing data
  - Out of sequence events

- Concurrency Control
  - Apply Optimistic Concurrency Control at Mongo
  - Optimise for collisions

Synchronisation & Concurrency

- Synchronisation
  - Out of sequence index instructions
  - elasticsearch external versioning
  - Can rebuild from scratch if need to
- Consistency
  - Which version is right?
  - Dates
  - Revision numbers from SQL
  - Independent updates

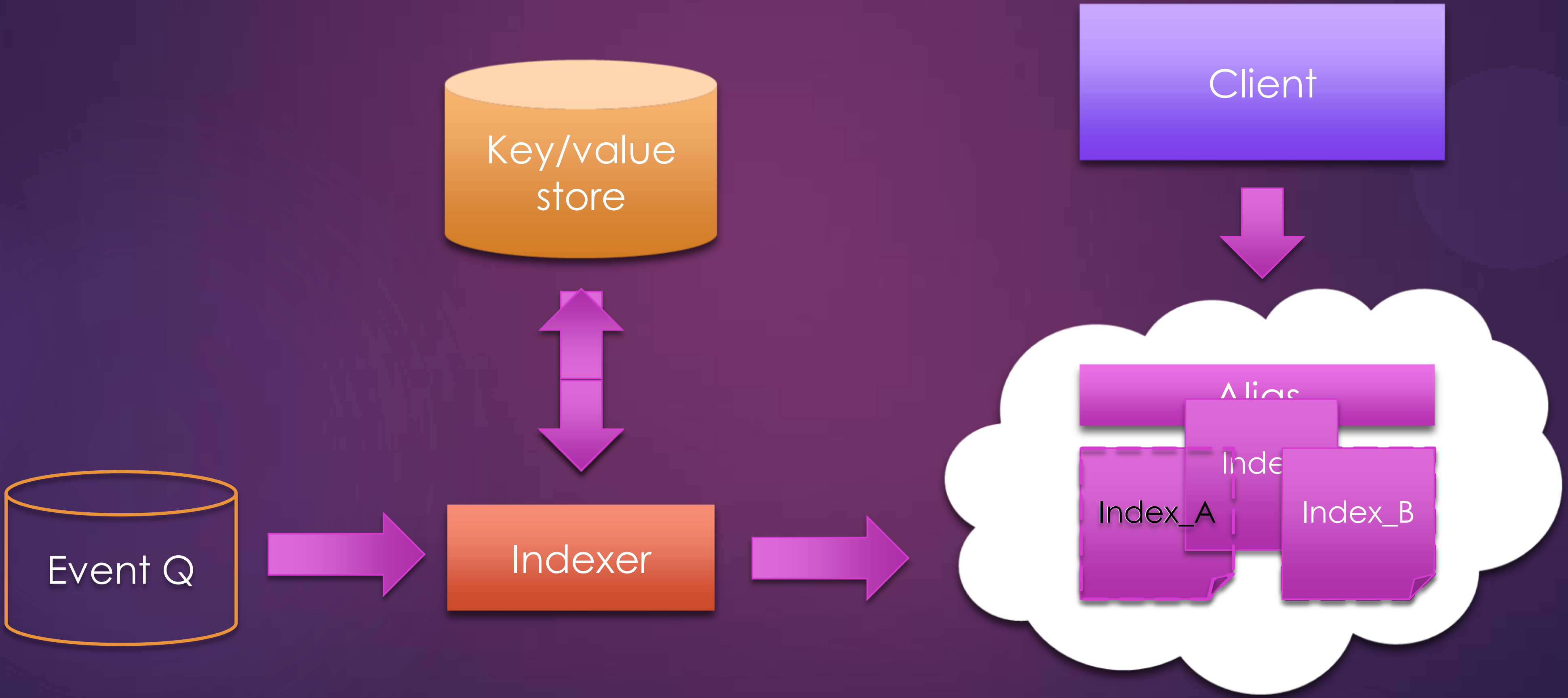
Figures

- Ingestion
  - 20m data points/day (continuously)
  - ~ 200GB
  - 3K msgs/second at peak

	Custom-Built Lucene	elasticsearch
Latency	3 hours	< 1 second
Bottleneck	Disk (SQL)	CPU

- Hardware
  - SQL: 2 x 12-core, 64GB, 72-spindle SAN
  - Indexing: 4 x 4-core, 8GB
  - Mongo: 1 x 4-core, 16GB, 1xSSD
  - Elastic: 5 x 4-core, 16GB, 1xSSD

Managing Change



Thanks

- @YannCluchey
- Concurrency Patterns with MongoDB
  http://slidesha.re/YFOehF
- Consistency without Consensus
  Peter Bourgon, SoundCloud
  http://bit.ly/1DUAO1R
- Eventually Consistent Data Structures
  Sean Cribbs, Basho
  https://vimeo.com/43903960