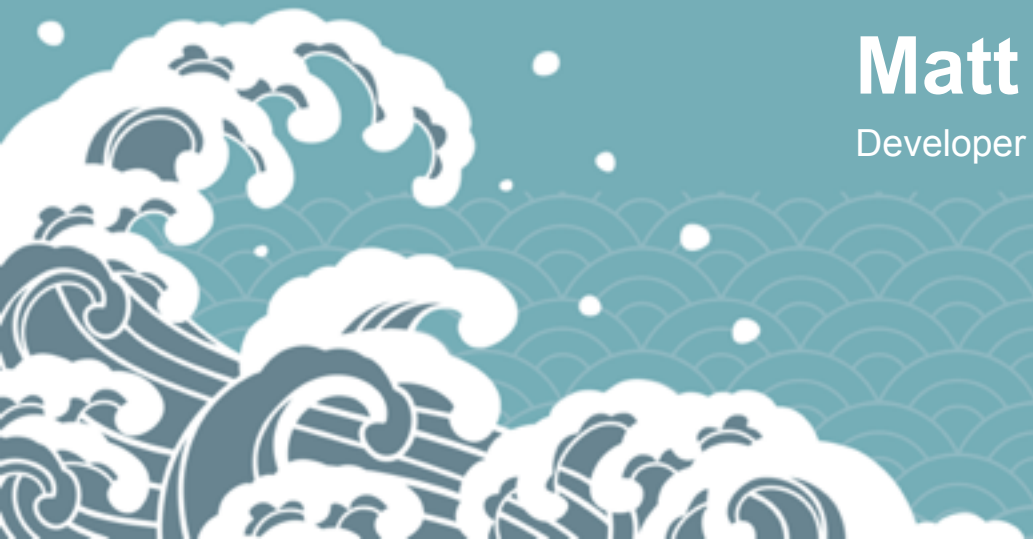
Designing for Distributed, Unstructured Data

Matt Brender

Developer Advocate at Basho



=> curl $RIAK/props

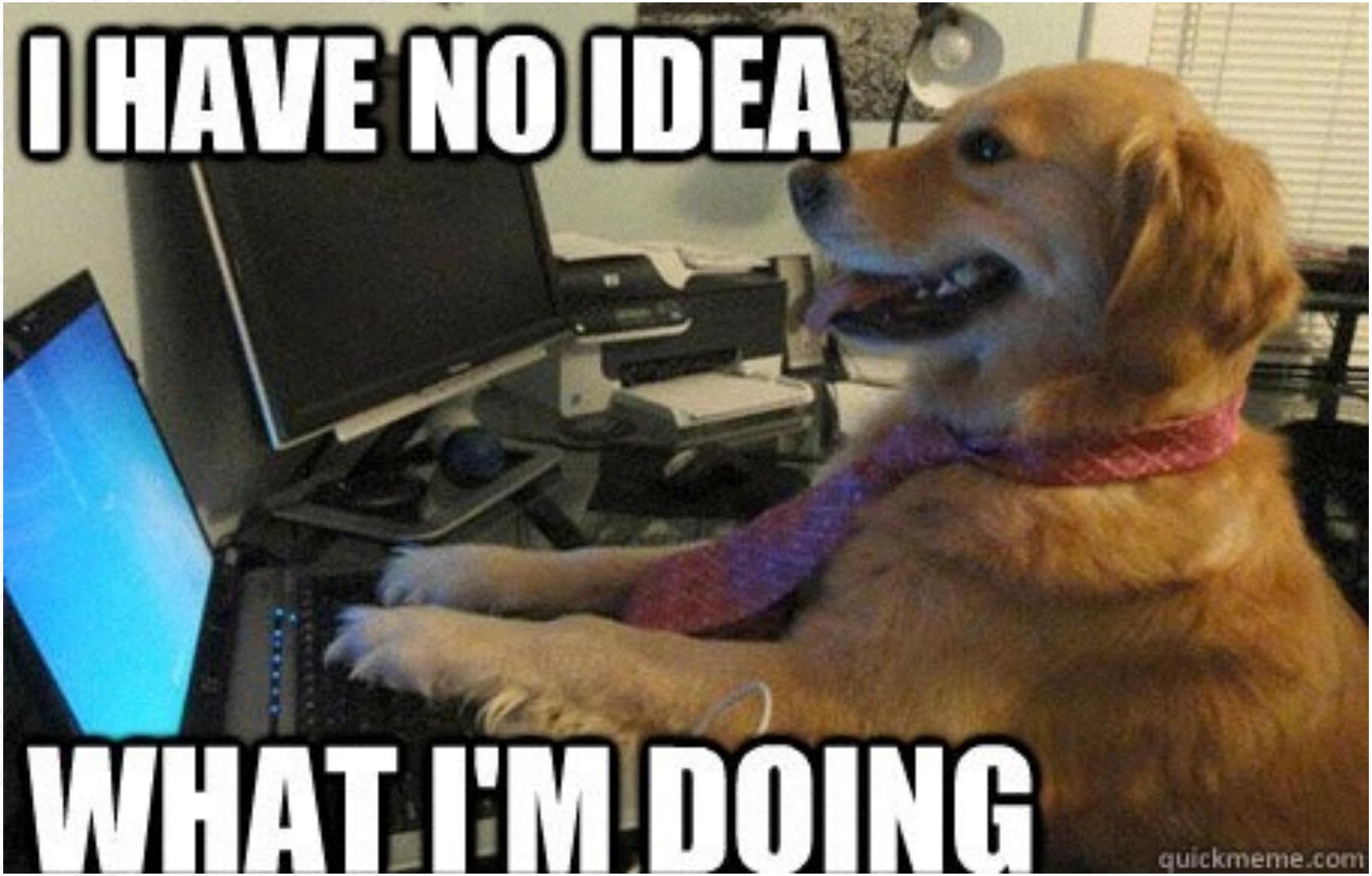
{ “Matt Brender” :

‘developer advocate’,
‘ops > dev’,
’mbrender@basho.com’,
‘@mjbrender’,
‘neckbeardinfluence.com’,
‘geek-whisperers.com’,
‘indoor enthusiast’

}

I'm saying "Riak"

Not "react," as in react.js


I HAVE NO IDEA
WHAT I'M DOING
quickmeme.com


ALERTLOGIC
COMPAREGROUP.I
LOOMIS | SAYLES
ROVIO
Virgin america
AURIC
SYSTEMS INTERNATIONAL
DELL
mochi
Tapjoy
McAfee
An Intel Company
NHS
UBER
bet365
dubizzle
statpro
Betzold
orange
T··Mobile·
EMERSON
temetra
XING
bunday
RIAK DEPLOYED WORLDWIDE
CenturyLink
Symantec.
Weborama
CISCO
Workshare
GT NEXUS
ERLANG
nordicmind
BEST BUY
OpenX
synacor
citymaps
Hibernum
ONE TWO TRIP!
Erlang SOLUTIONS
VOXER
KIVRA
ideeli
innometrics
IDC Frontier
TURNER
YAHOO! JAPAN
comcast
G2 We Speak Gaming
klarna
RIOT GAMES
The Weather Channel
TRIFORK ...think software
yammer

dataintensive.net

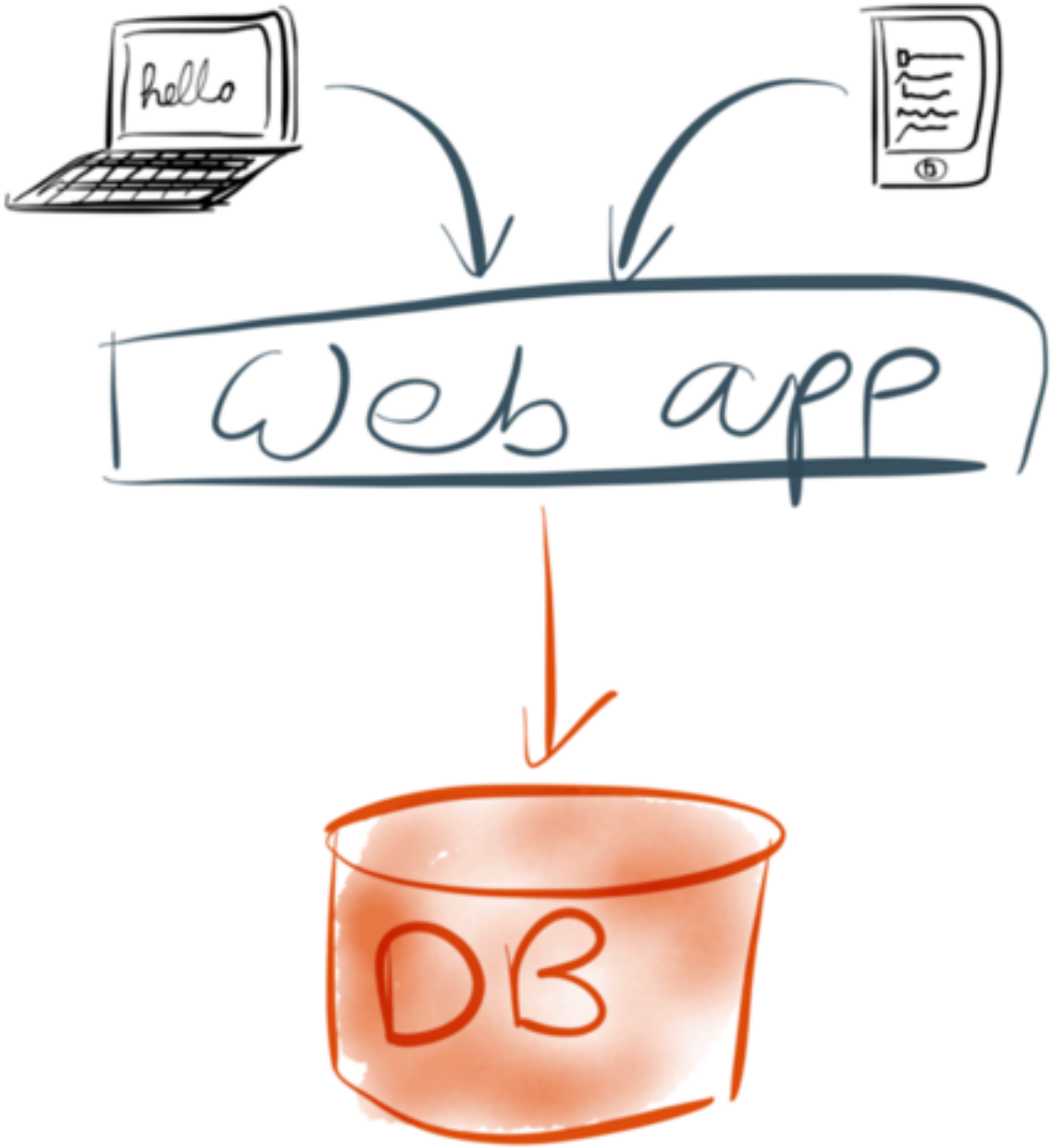
@martinkl

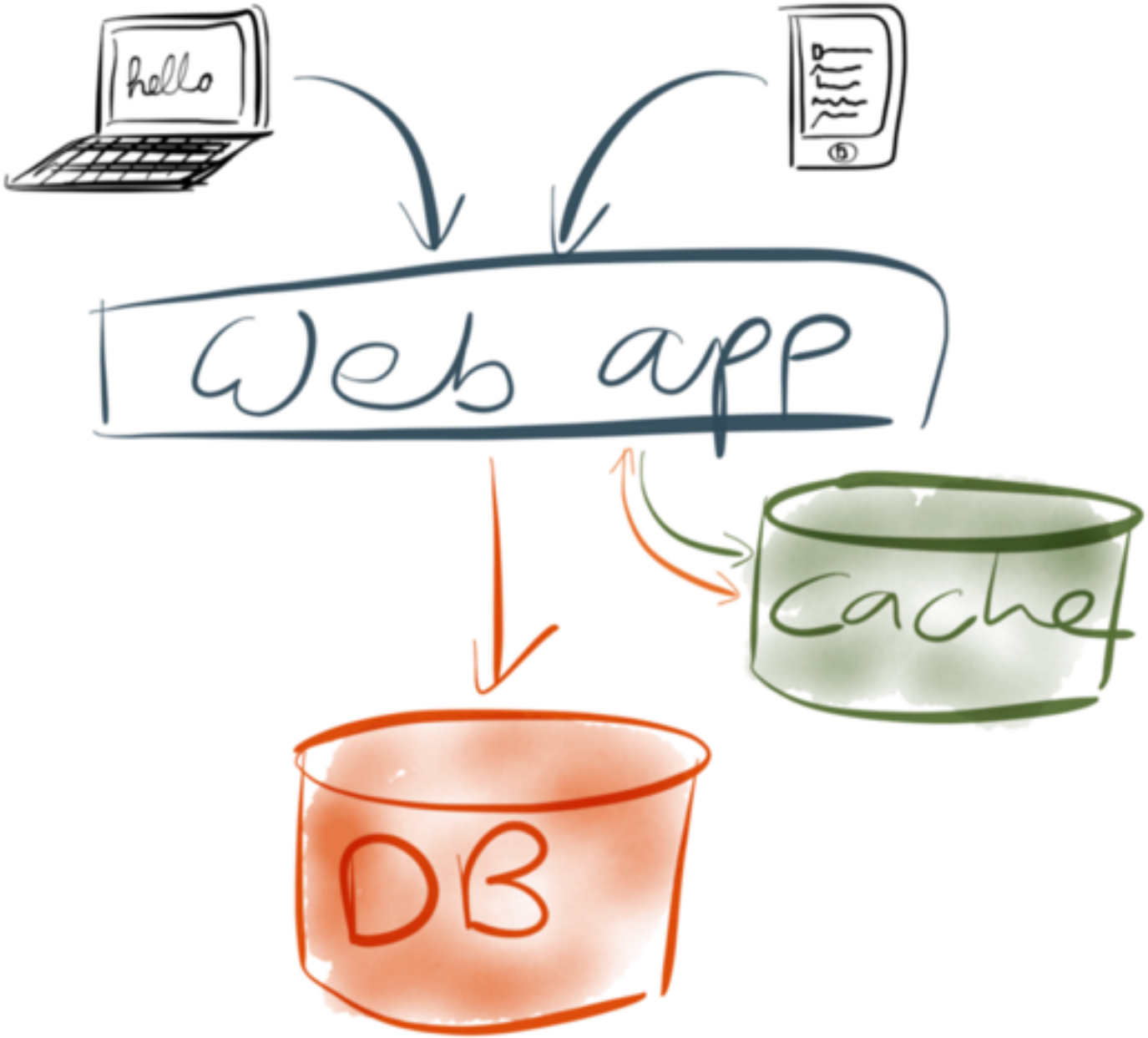
```
{
  "text": ”Woot! #qconnewyork",
  "entities": {
    "hashtags": [“#qconnewyork”],
    "symbols": [],
    "urls": [],
    "user_mentions": [{
      "screen_name": ”mjbrender",
      "name": ”Matt Brender",
      "id": 4948123,
      "id_str": ”42424242",
      "indices": [81, 92]
    }, {
      "screen_name": ”mjbrender",
      "name": ”Matt Brender",
      "id": 376825877,
      "id_str": "376825877",
      "indices": [121, 132]
    }]
  }
}
```

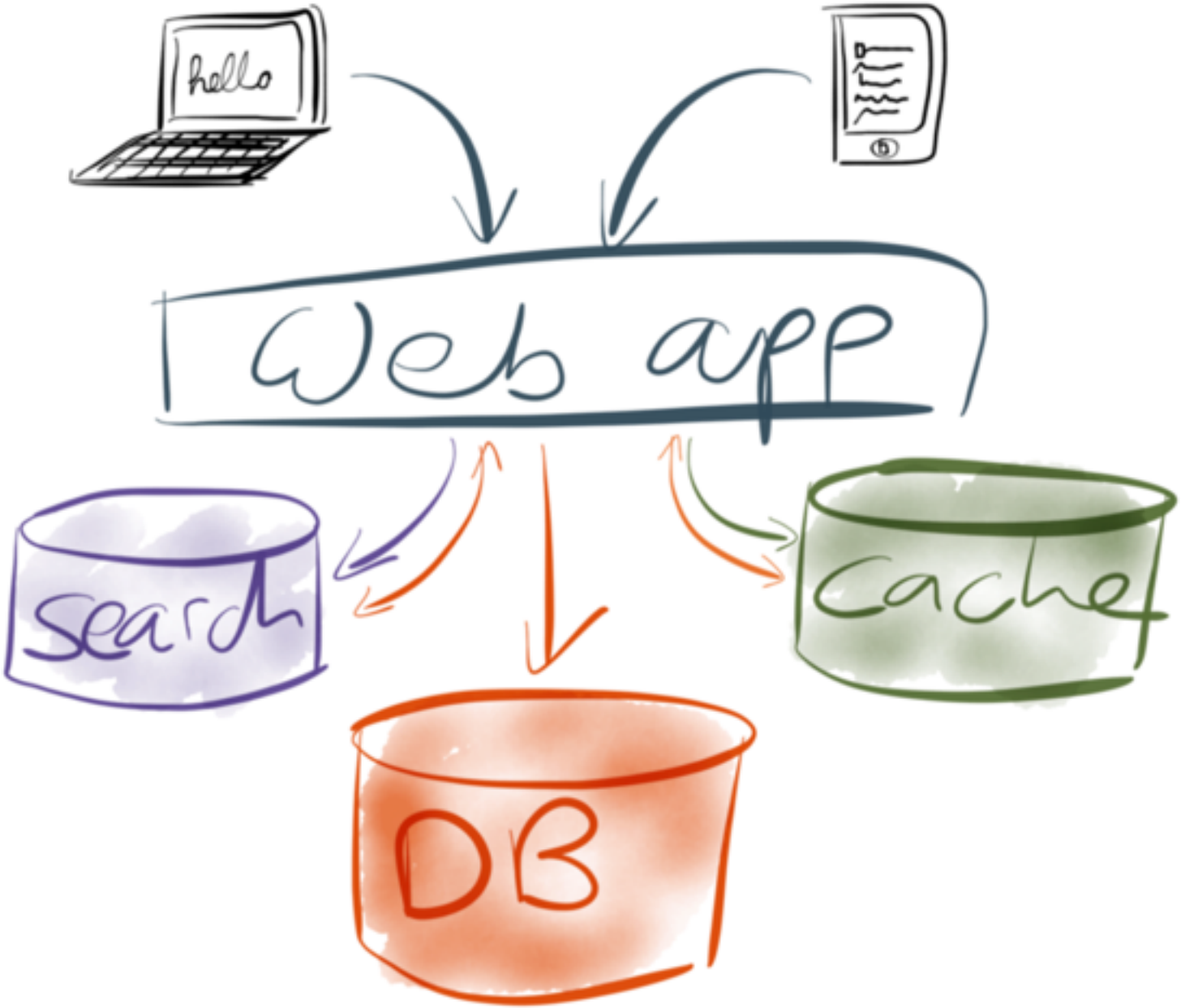
Just Hording?

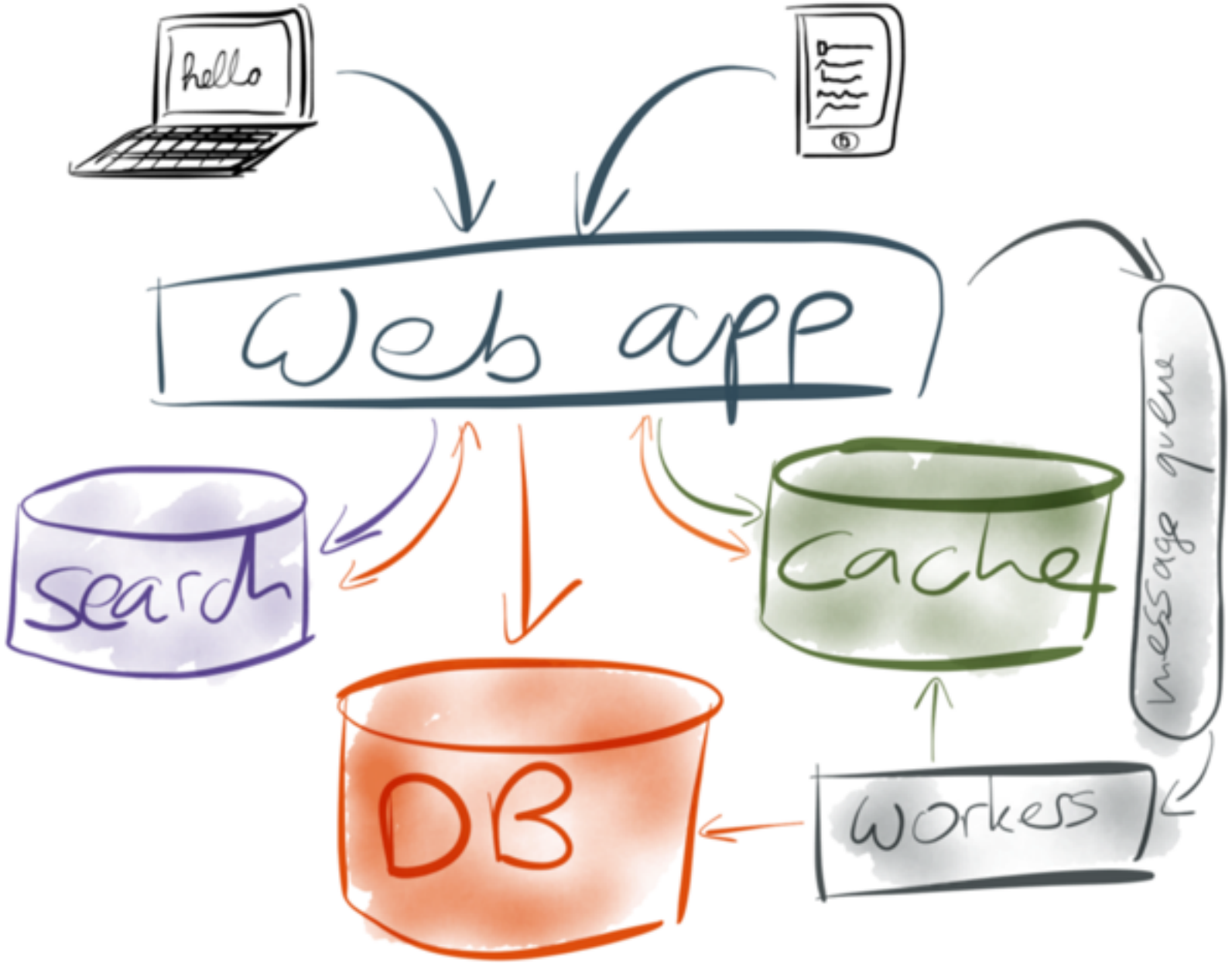
Just Hording?

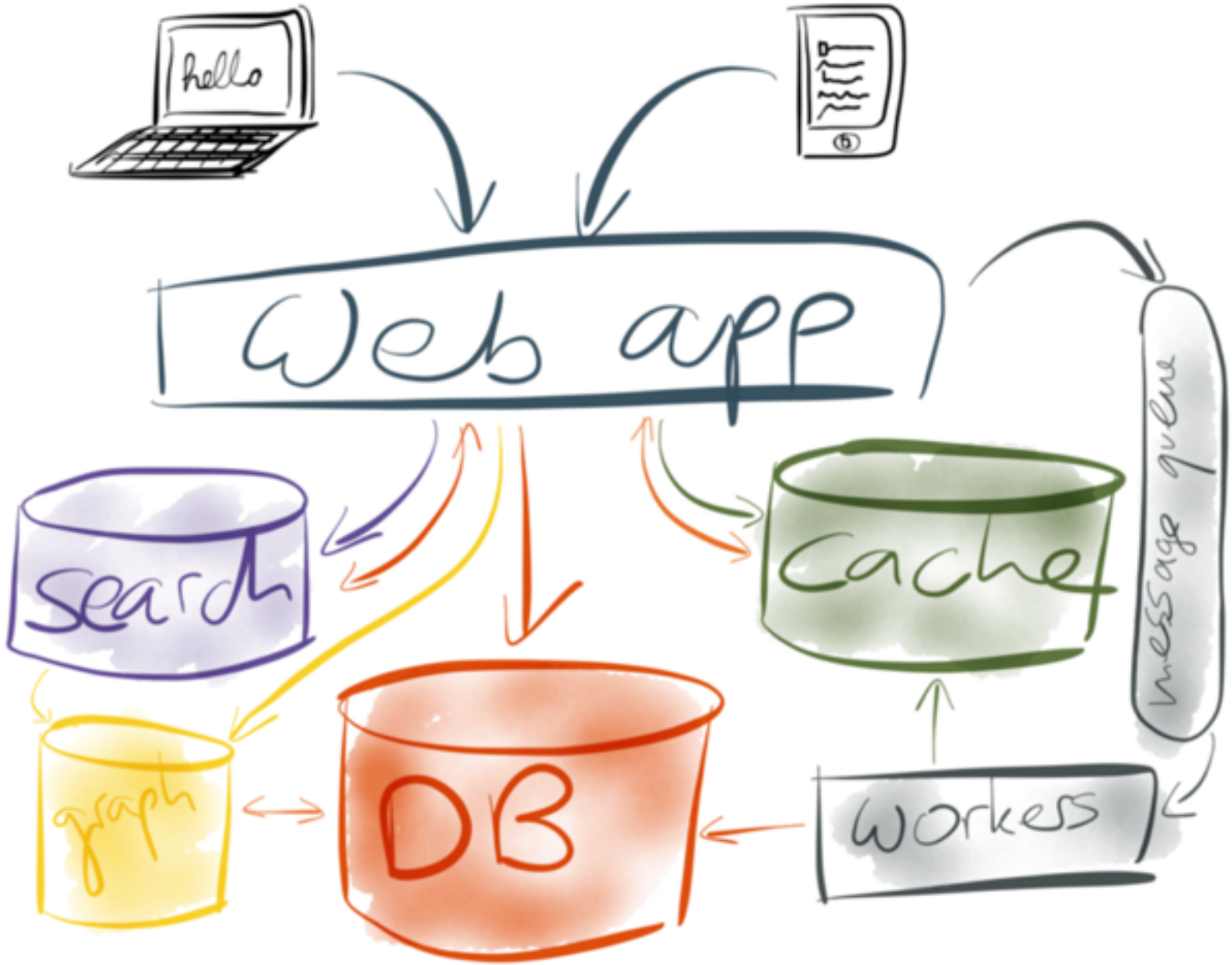
A common pattern

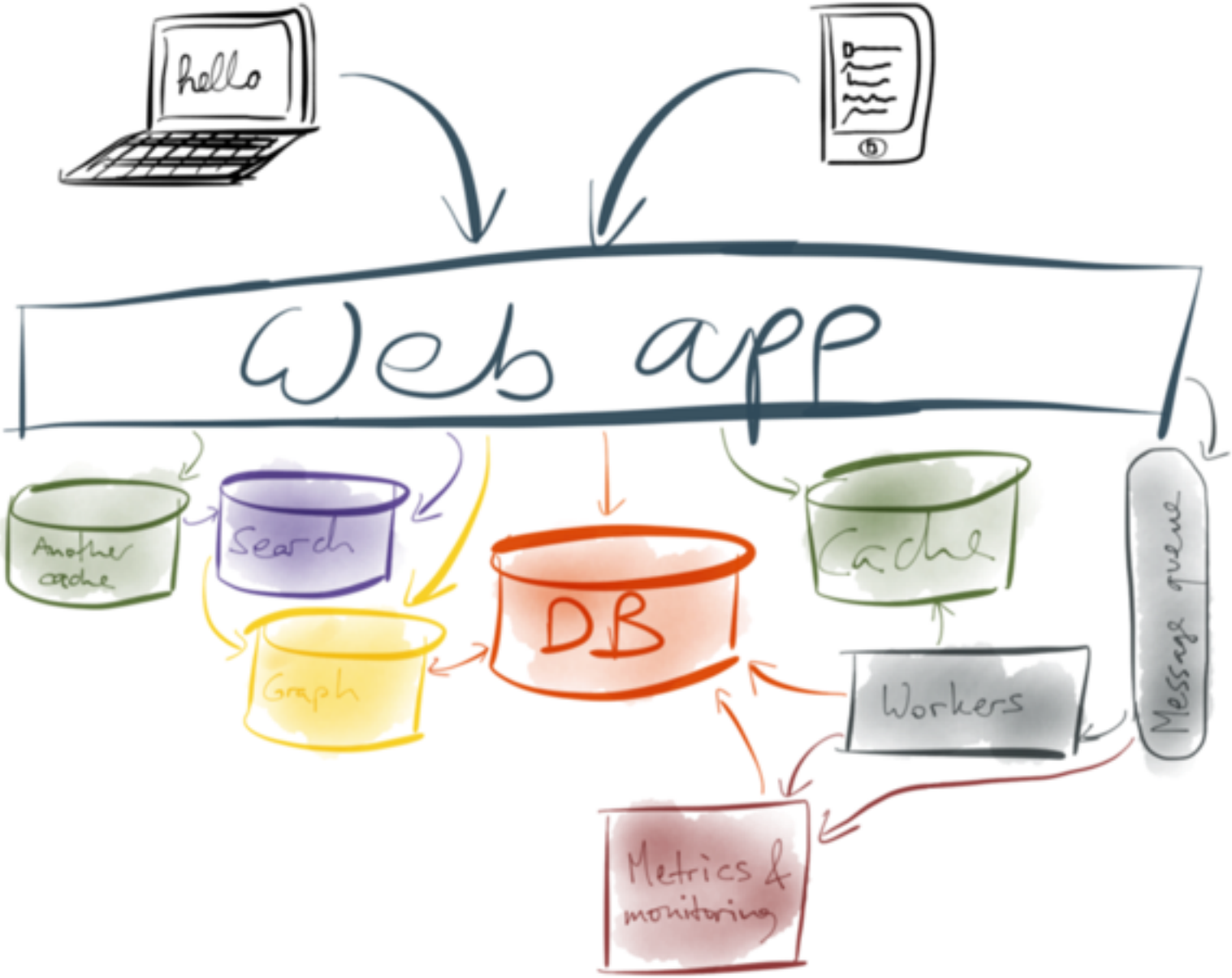

hello
Web app
DB

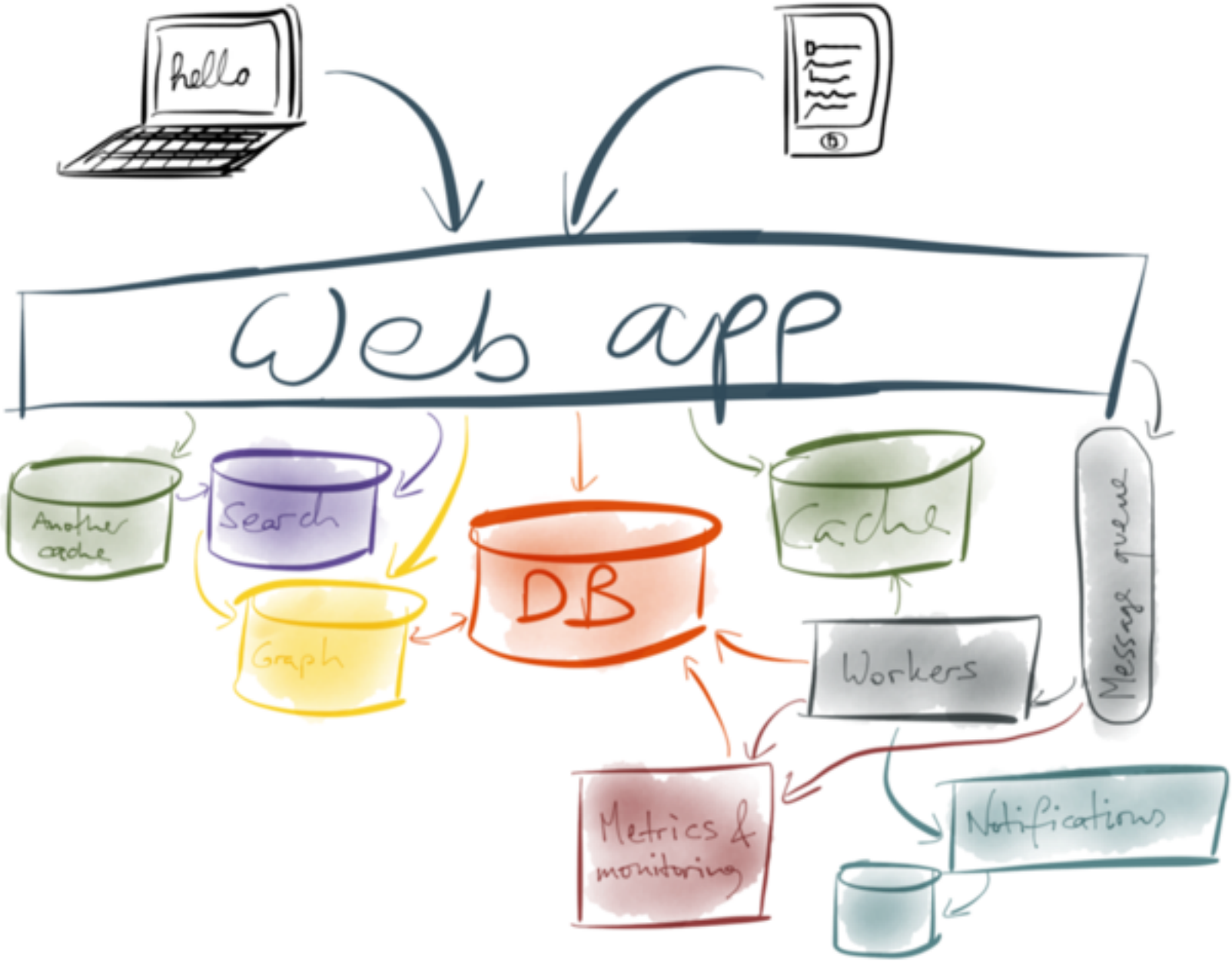

hello
Web app
cache
DB

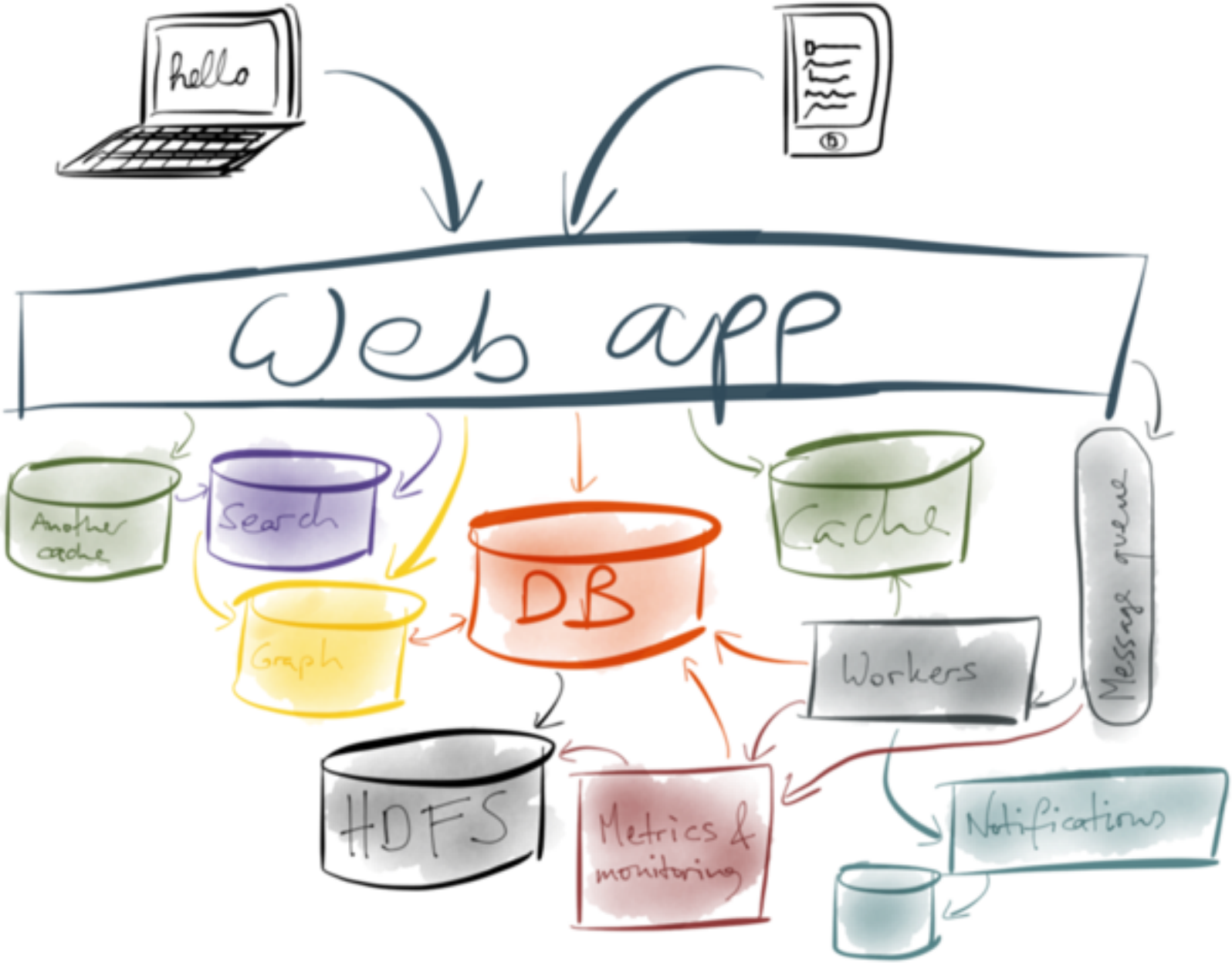

hello
Web app
search
cache
DB

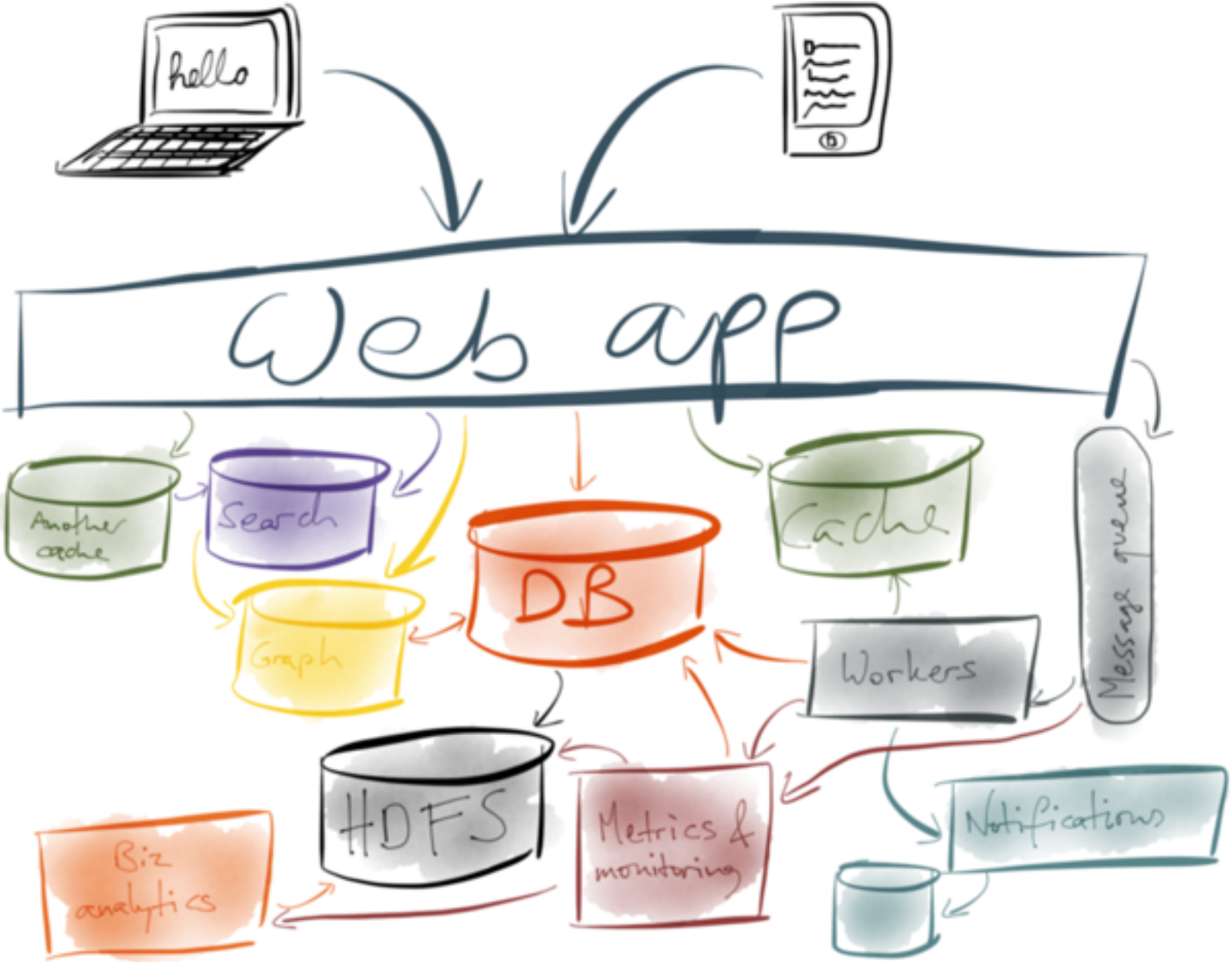

hello
Web app
search
cache
message queue
DB
workers


hello
Web app
search
cache
message queue
graph
DB
workers


hello
Web app
Another cache
Search
Graph
DB
Cache
Message queue
Workers
Metrics & monitoring


hello
Web app
Another cache
Search
Graph
DB
Cache
Message queue
Workers
Metrics & monitoring
Notifications


hello
Web app
Another cache
Search
Graph
DB
Cache
Message queue
Workers
HDFS
Metrics & monitoring
Notifications


hello
Web app
Another cache
Search
Graph
DB
Cache
Message queue
Workers
HDFS
Metrics & monitoring
Biz analytics
Notifications


hello
Web app
Another cache
Search
Index builder
Biz analytics
HDFS
Metrics & monitoring
Workers
Message queue
Notifications

Our Problem(s)

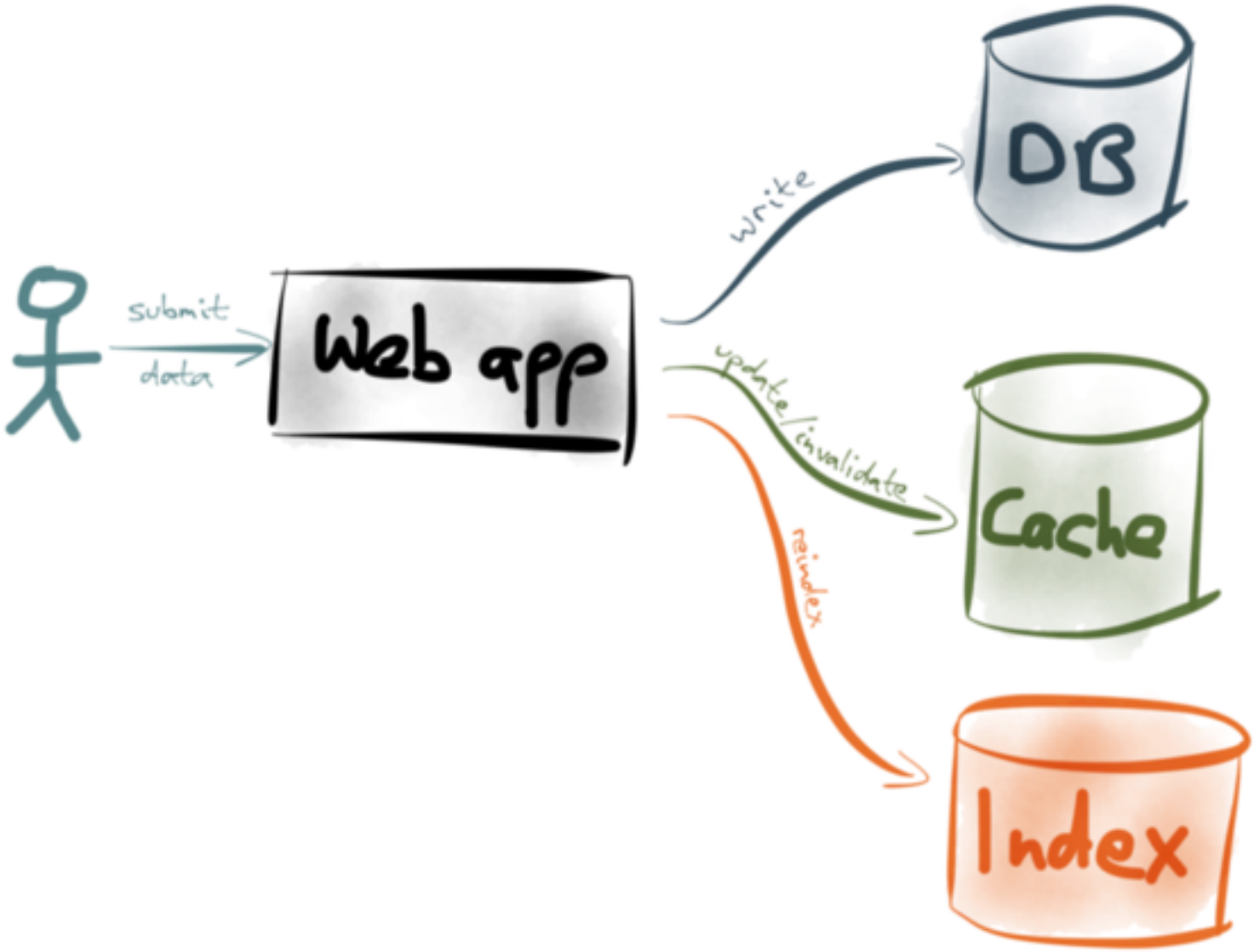
- **Same data in different formats**
  - Cache
  - Denormalisation
  - Indexes
  - Aggregations

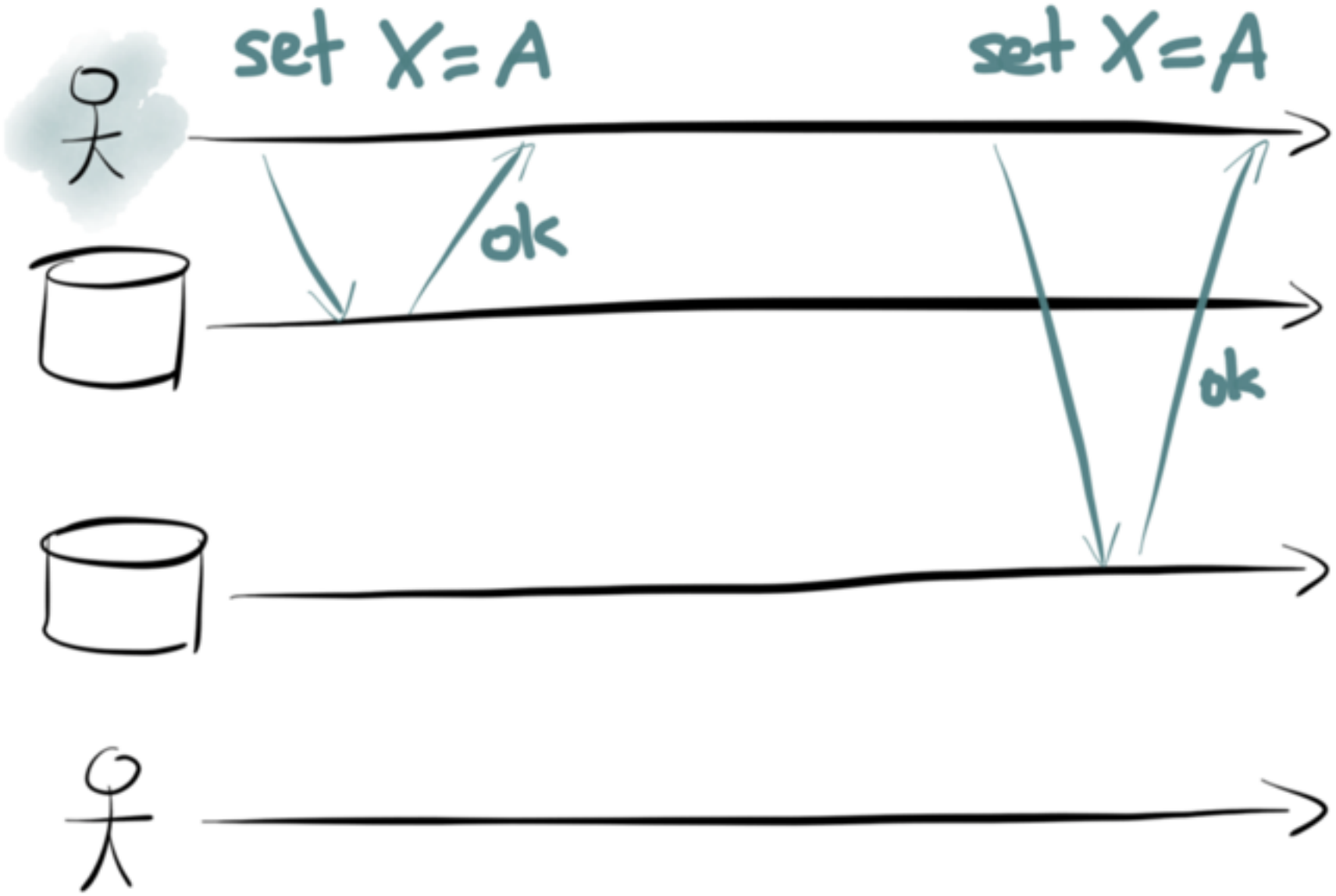
- **We're sticking to what we know**
  - Relational databases with SQL queries
  - Not anticipating scaling needs

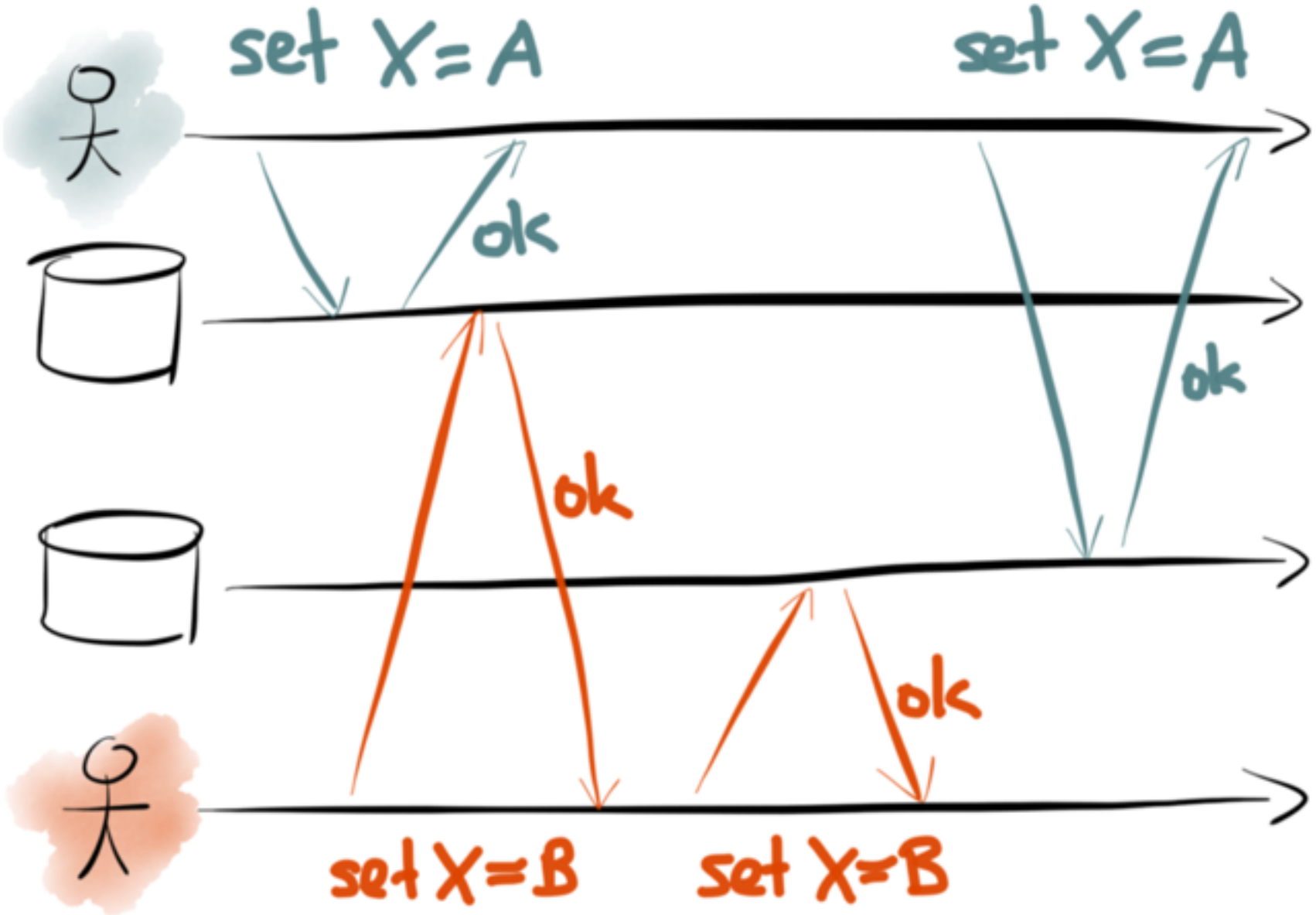
- **We're not sure what's next**
  - Bitten by architectural choices in the past
  - New systems require consideration
  - Not sure what's justifies investment

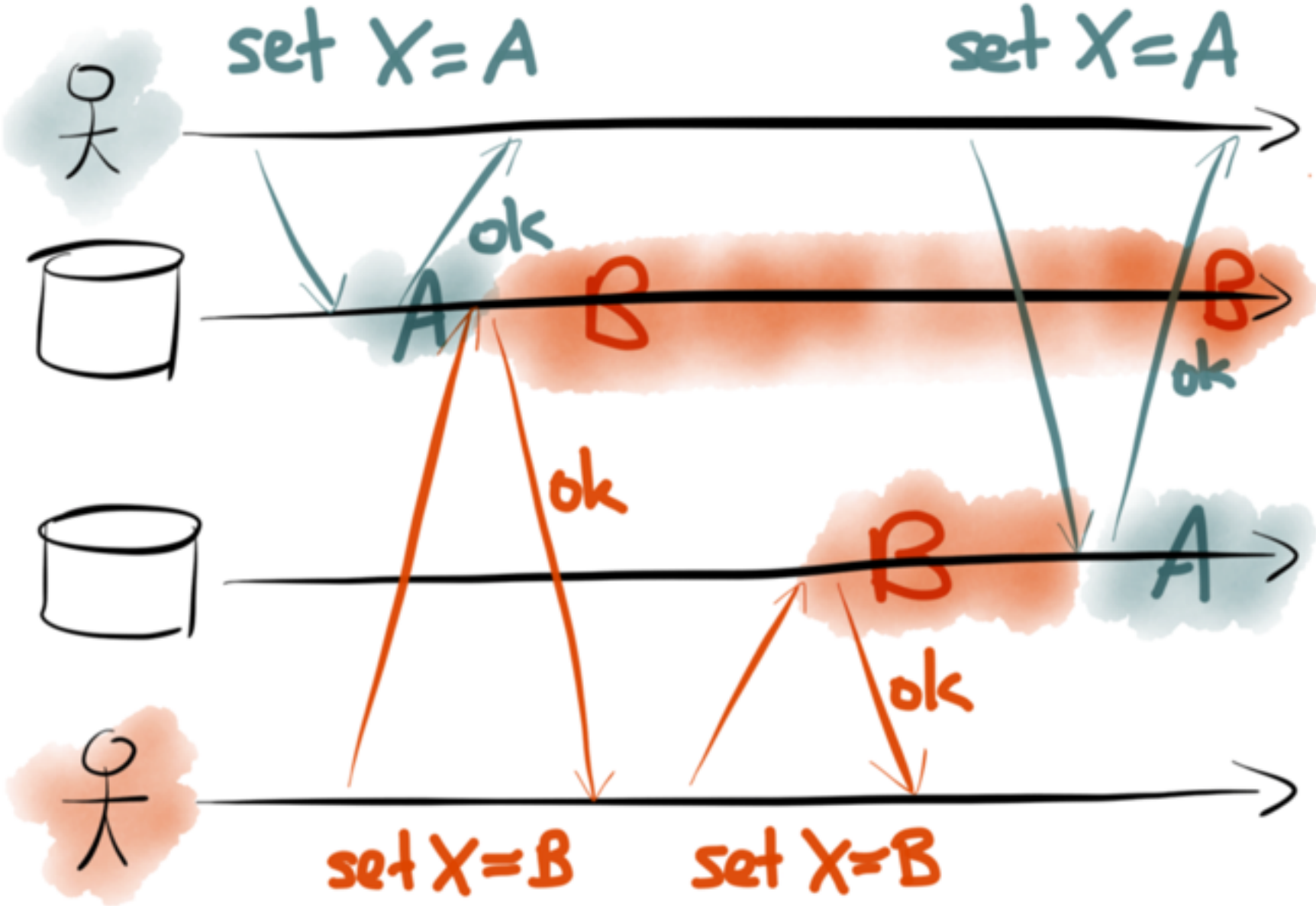
Can’t I just…

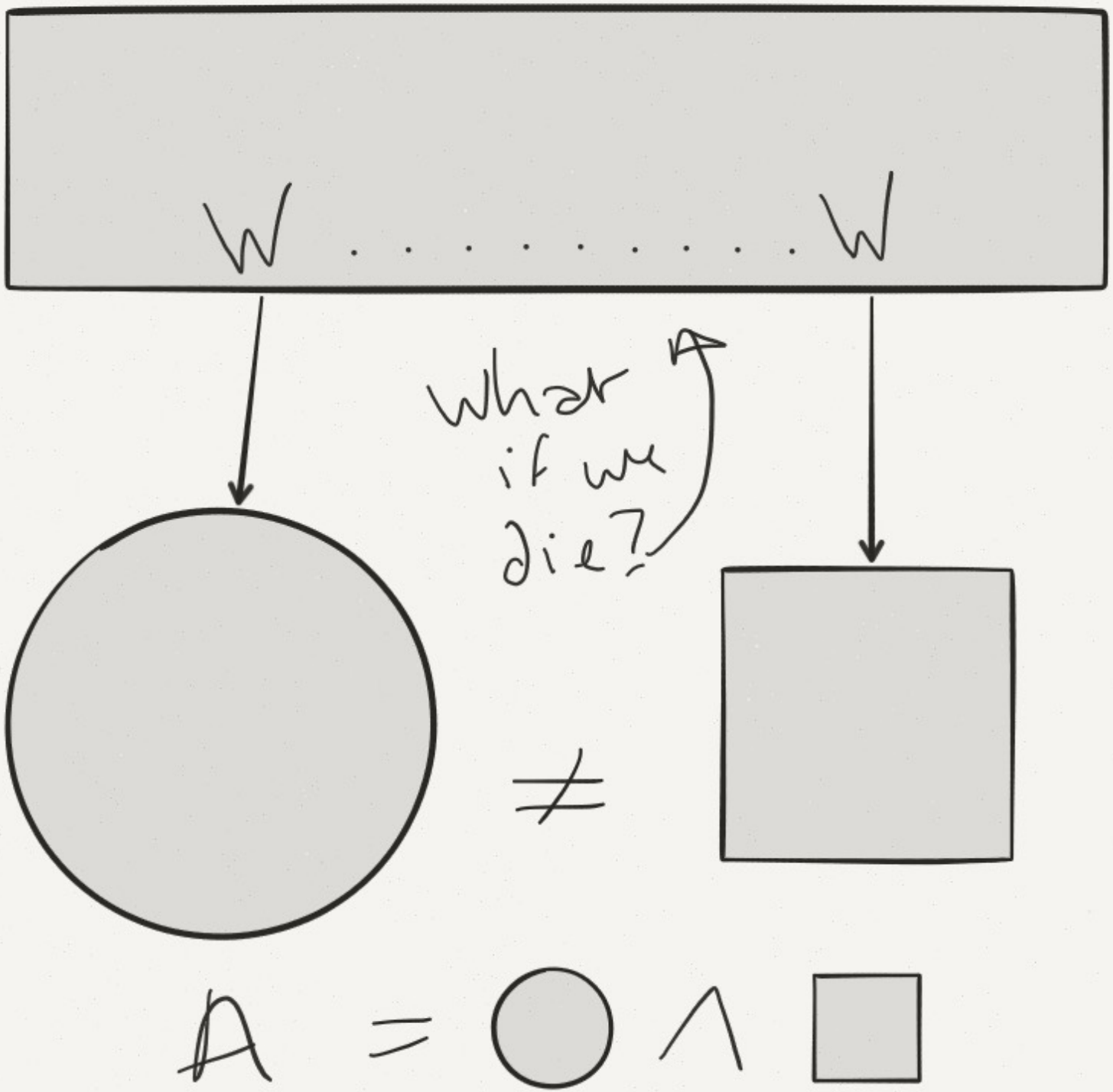
"Making sure the data ends up in all the right places"

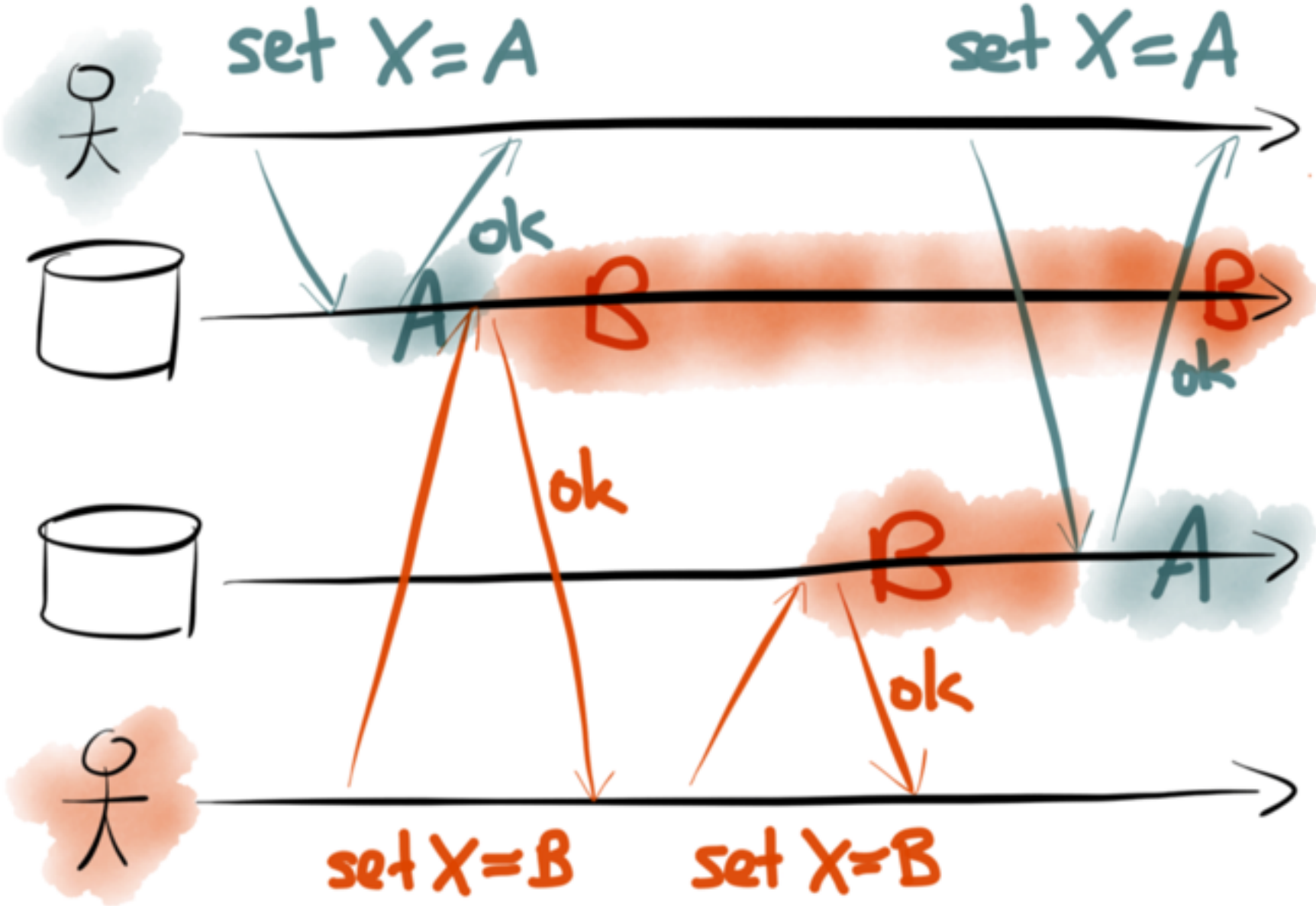

submit
data
Web app
write
DB
update/invalidate
Cache
reindex
Index


set X=A
set X=A
ok
ok


set X=A
set X=A
ok
ok
ok
ok
set X=B
set X=B


set X=A
set X=A
ok
A
B
B
ok
ok
B
A
ok
set X=B
set X=B


W . . . . . W
what if we die?
≠
A = ∧


set X=A
set X=A
ok
A
B
B
ok
ok
B
A
ok
set X=B
set X=B

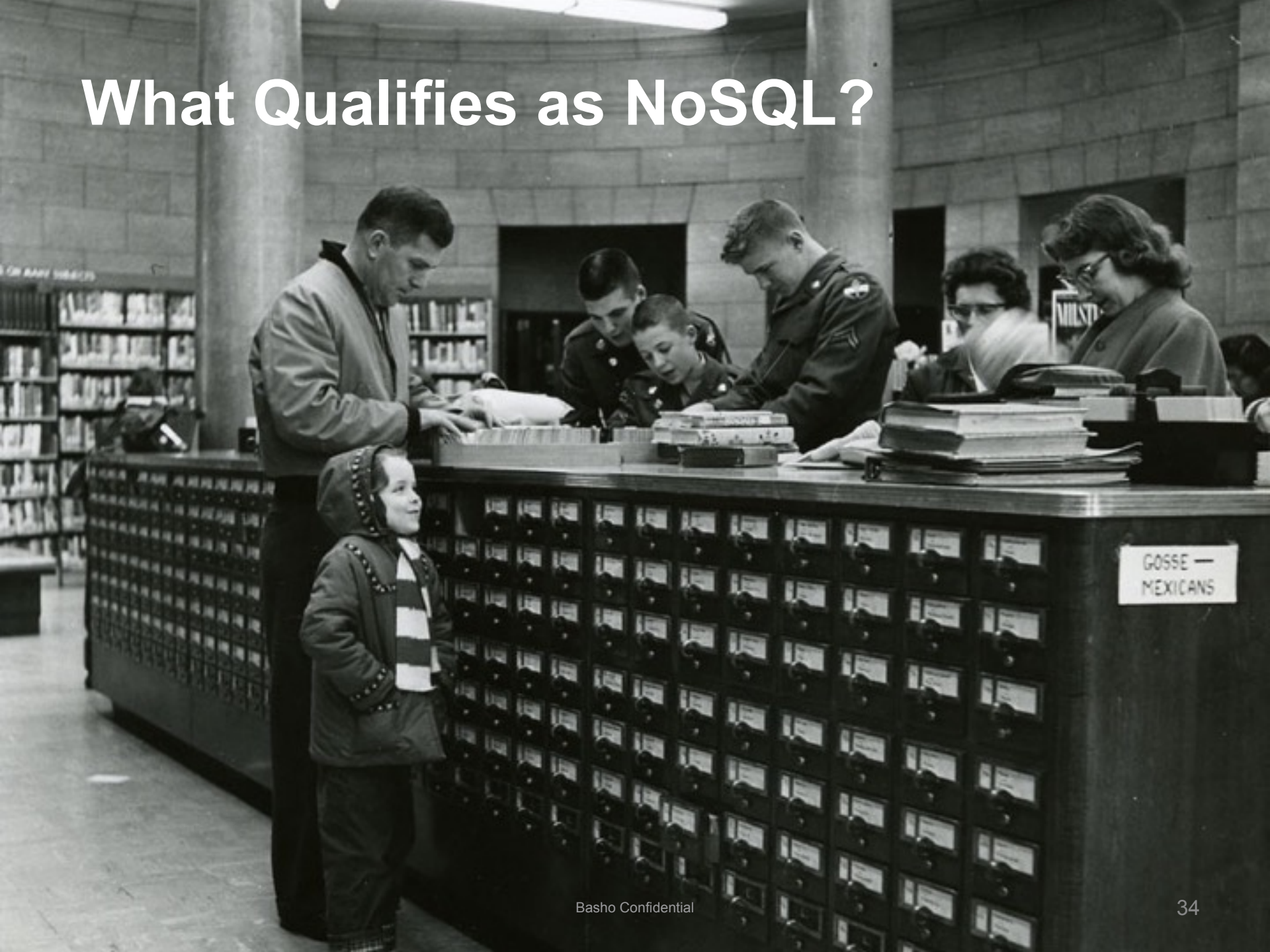
The Choices

This or That

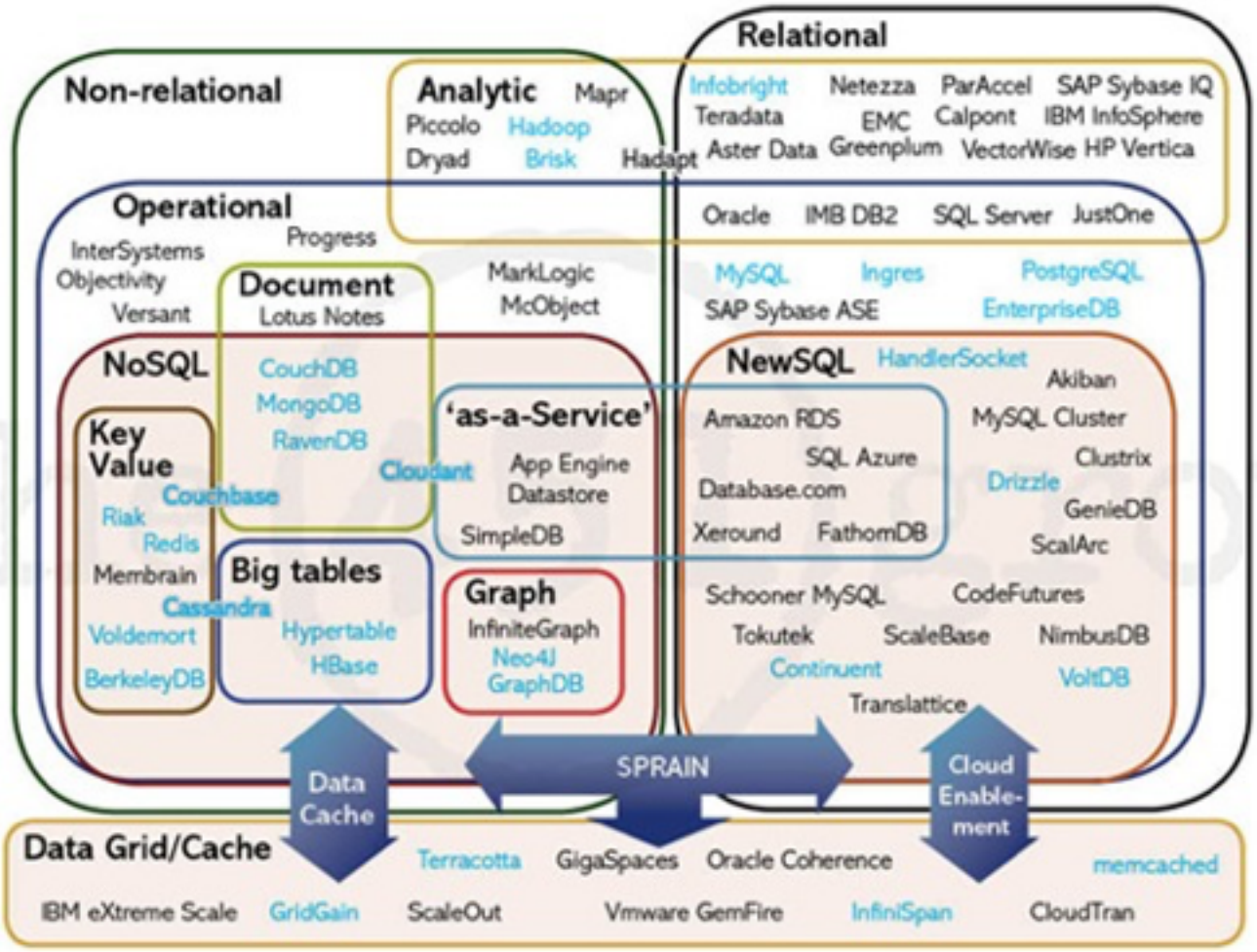
- **NoSQL**
  - Types
    - Key/Value
    - Document
    - Columnar
    - Graph
- **“Messaging Queues”**
  - Pub/Sub
  - Commit Log
- **Hadoop**
  - HDFS
  - Map/Reduce
  - YARN
- **Spark**
  - Successor to Map/Reduce
  - Compute-focused

So, NoSQL

What Qualifies as NoSQL?



NOSQL Community



Persistence

Querying

Scaling

Persistence

redis

riakKV

Querying

Other Queries

Understanding how you get your data back

Query Languages

- SQL(?)

Query Interfaces

- HTTP/S
- Protocol Buffers

Apache Solr Integration

Write it like Riak. Query it like Solr.

Distributed Full-Text Search

Standard full-text Solr queries automatically expand into distributed search queries for a complete result set across instances.

Ad-Hoc Query Support

Broad support for Solr query parameters, e.g., exact match, range queries, and/or/not, sorting, pagination, scoring, ranking, etc.

Index Synchronization

Data is automatically synchronized between Riak KV and Solr using intelligent monitoring to detect changes, and propagates those to Solr indexes.

Solr API Support

Query data in Riak KV using existing Solr APIs

Auto-Restart

Monitor Solr OS processes continuously and automatically start or restart them whenever failures are detected.

Polylingual Querying

There are a diverse group of client libraries for Riak that support both the HTTP and Protocol Buffer APIs:

Basho Supported Libraries:

- Java
- Ruby
- Python
- PHP
- Erlang
- .NET
- Node.js
- C

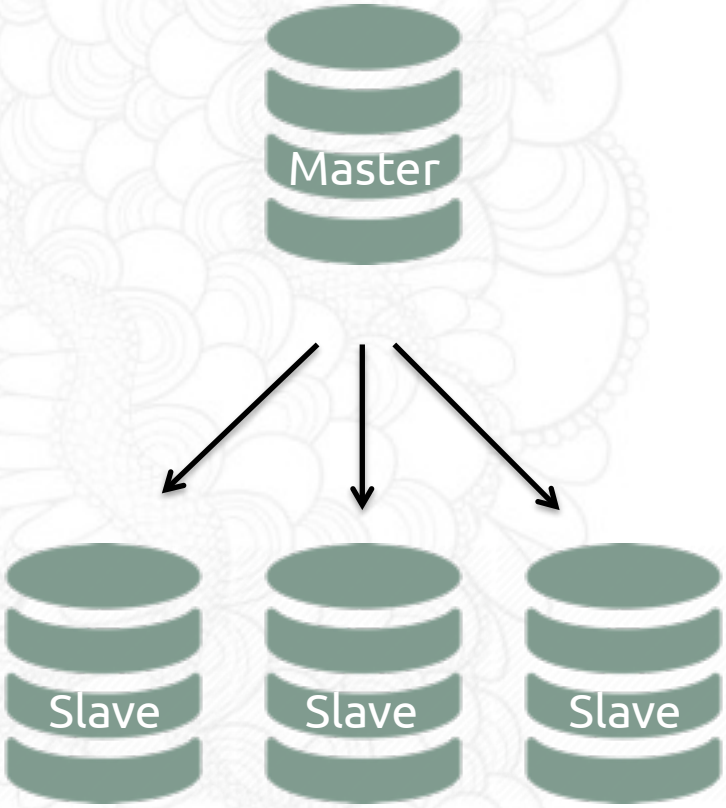
Community Libraries:

- Clojure
- Go
- Perl
- Scala
- R

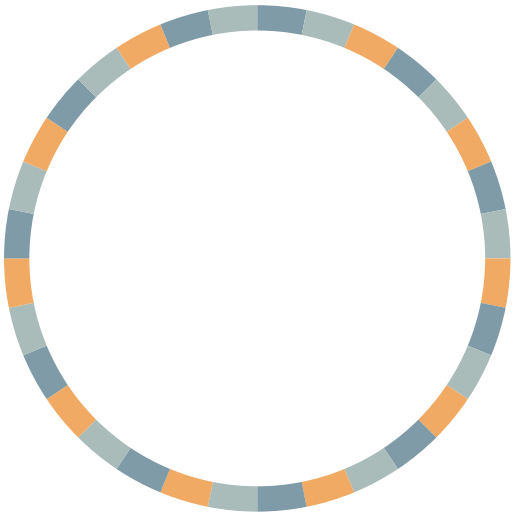
Scale means

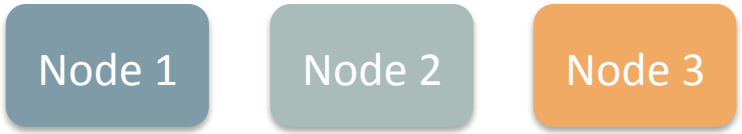
Sharding

Sharding Strategies

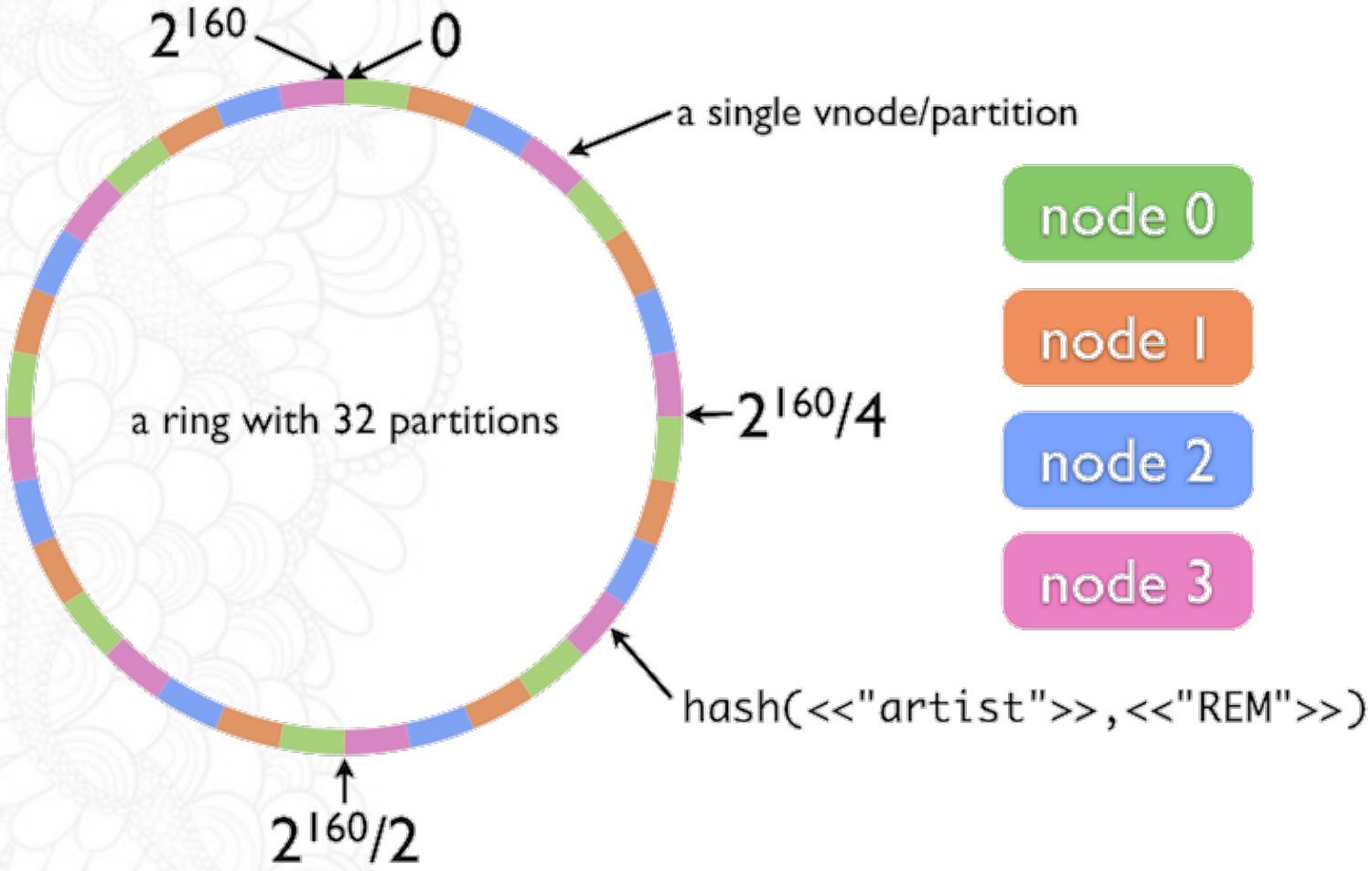


OR





Sharding Strategies



CAP Theorem

CA
RDBMS
MySQL
Postgres

AP
Riak
Cassandra
Couchbase
Voldemort

CP
MongoDB
Redis
BigTable
Hbase

What Are You Sacrificing?

- **CA**
  - Data is consistent and R/W from any node until partition, when data will be out of sync (and won't re-sync)
- **CP**
  - Data is consistent between all nodes, and maintains partition tolerance (preventing data de-sync) by becoming unavailable when a node goes down
- **AP**
  - Nodes remain online even if they can't communicate with each other and will resync data once the partition is resolved, but you aren't guaranteed that all nodes will have the same data (either during or after the partition)

The Dynamo Paper

Dynamo: Amazon's Highly Available Key-value Store

Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall and Werner Vogels

Amazon.com

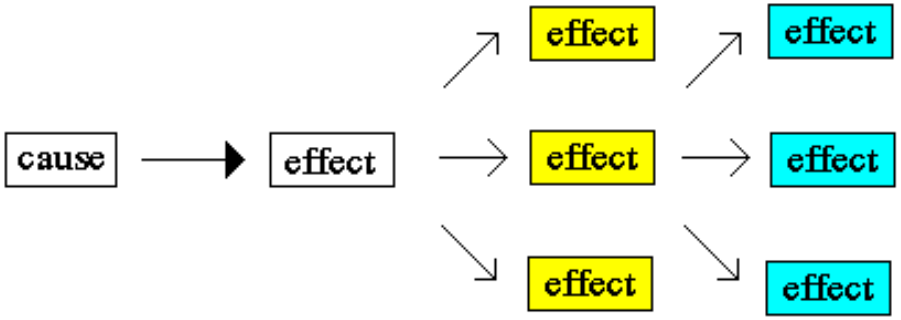
ABSTRACT

Reliability at massive scale is one of the biggest challenges we face at Amazon.com, one of the largest e-commerce operations in the world; even the slightest outage has significant financial consequences and impacts customer trust. The Amazon.com platform, which provides services for many web sites worldwide, is implemented on top of an infrastructure of tens of thousands of servers and network components located in many datacenters around the world. At this scale, small and large components fail continuously and the way persistent state is managed in the face of these failures drives the reliability and scalability of the software systems.

One of the lessons our organization has learned from operating Amazon's platform is that the reliability and scalability of a system is dependent on how its application state is managed. Amazon uses a highly decentralized, loosely coupled, service oriented architecture consisting of hundreds of services. In this environment there is a particular need for storage technologies that are always available. For example, customers should be able to view and add items to their shopping cart even if disks are failing, network routes are flapping, or data centers are being destroyed by tornados. Therefore, the service responsible for managing shopping carts requires that it can always write to and read from its data store, and that its data needs to be available across multiple data centers.

Conflict

Conflict Resolution



set conflict resolution

2015:05:25

```
{

["Tom" : "Beth"],
["Beth" : "Tom"],
["George" : "Jim"]

}
```

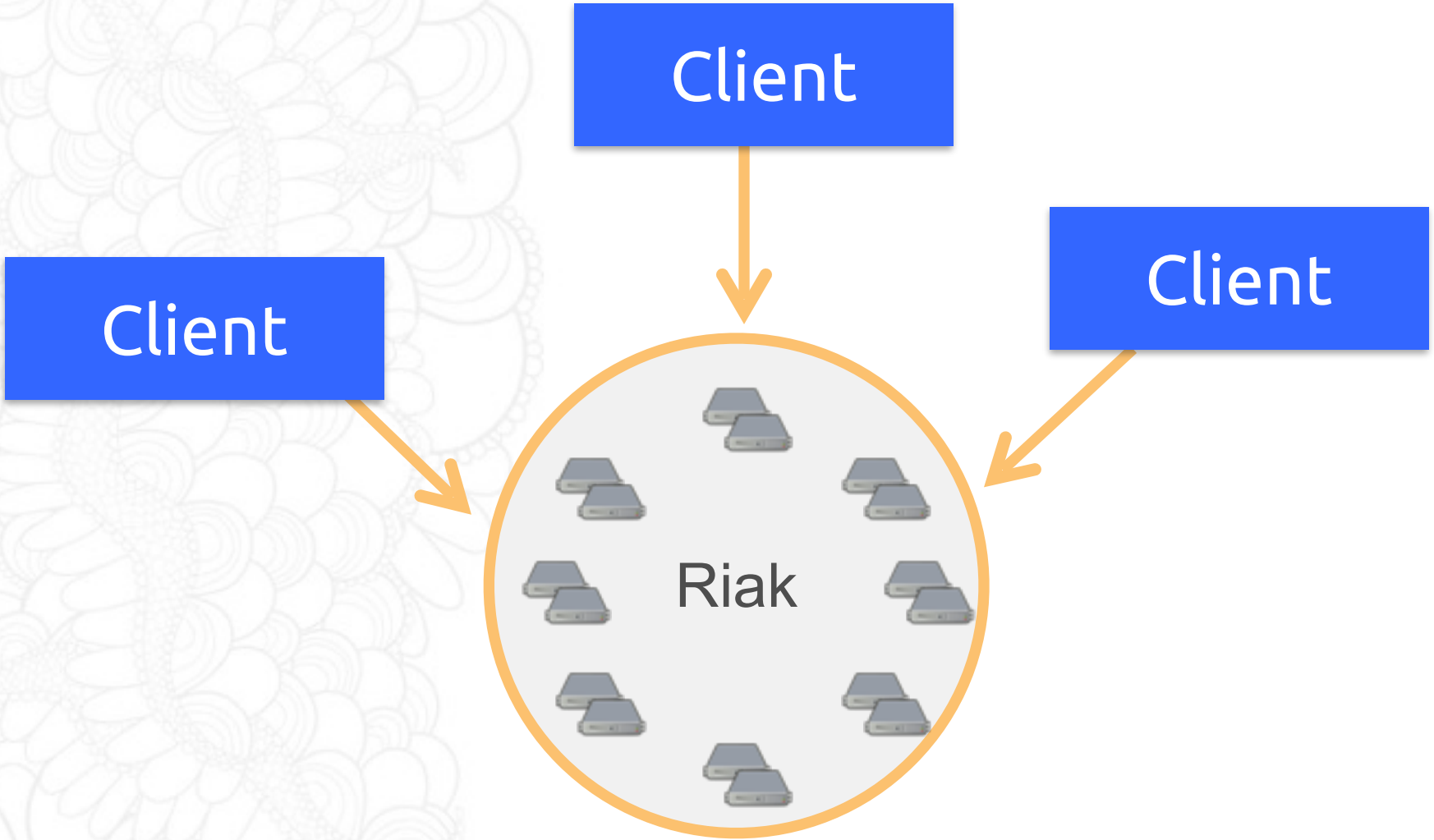
2015:05:26

```
{

["George" : "Tom"],
["Beth" : "Jim"],
["George" : "Jim"]

}
```

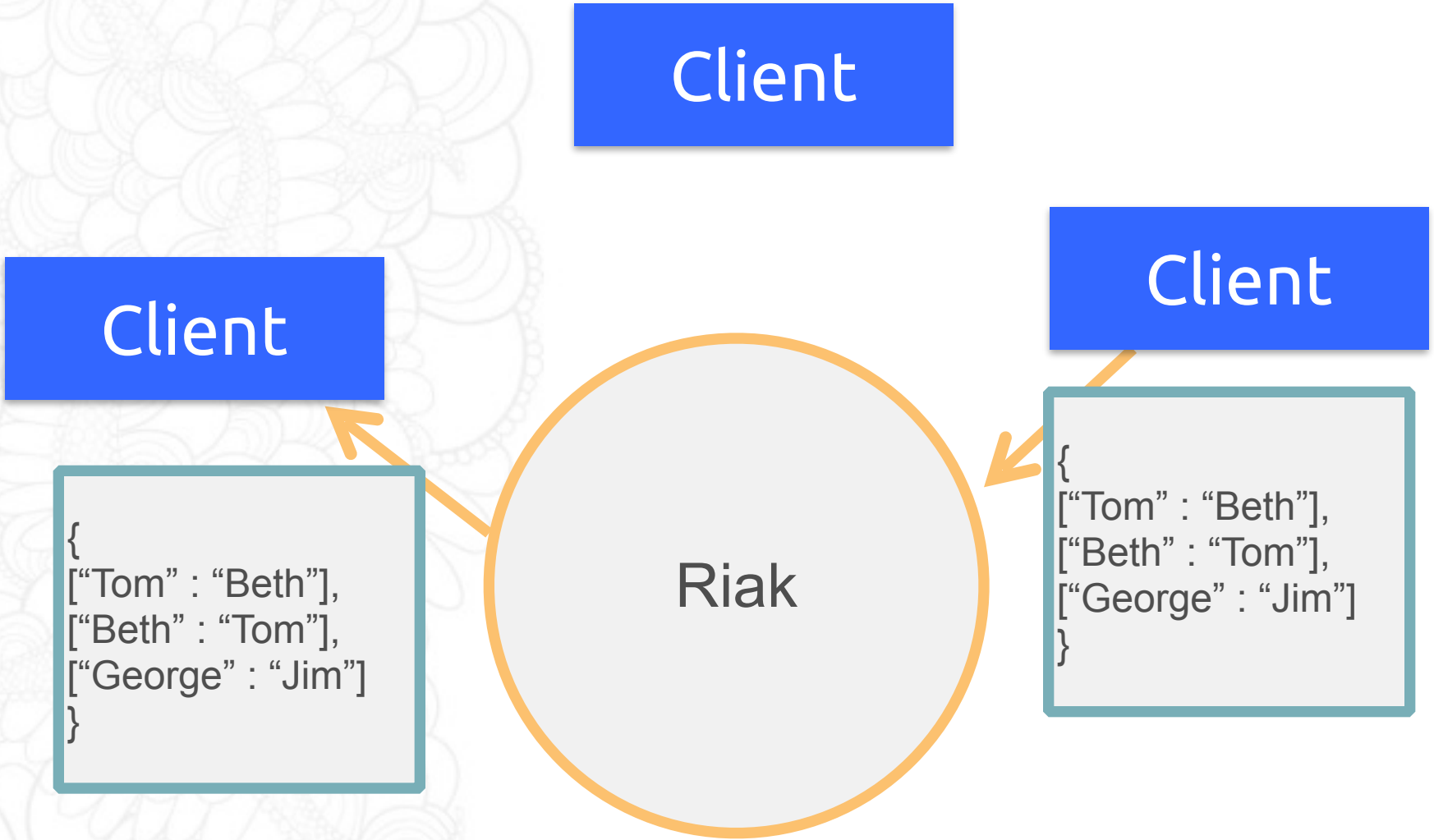
2015:05:27

```
{

["Beth" : "Tom"],
["Beth" : "Jim"],
["Beth" : "George"]

}
```

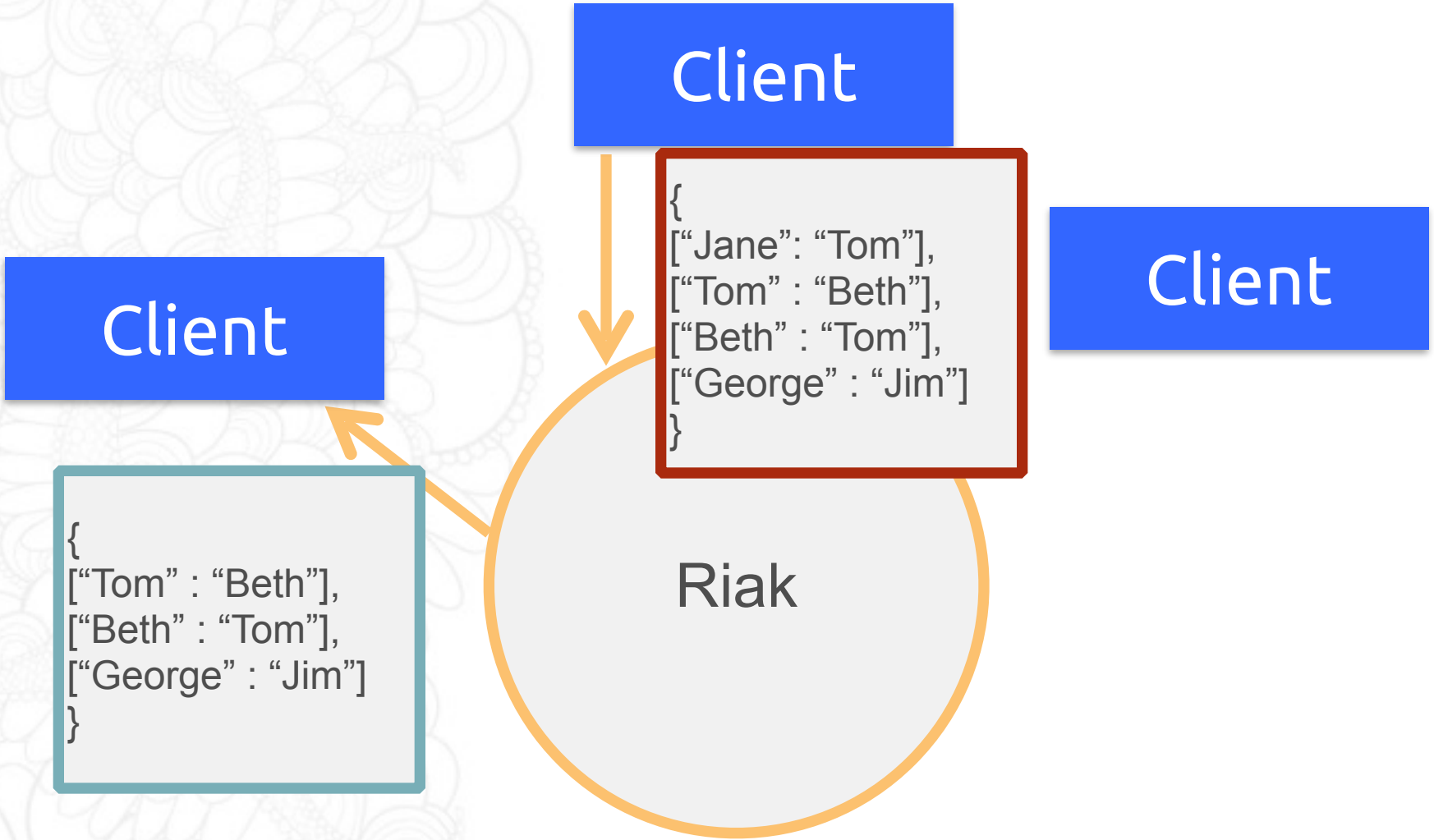
set conflict resolution



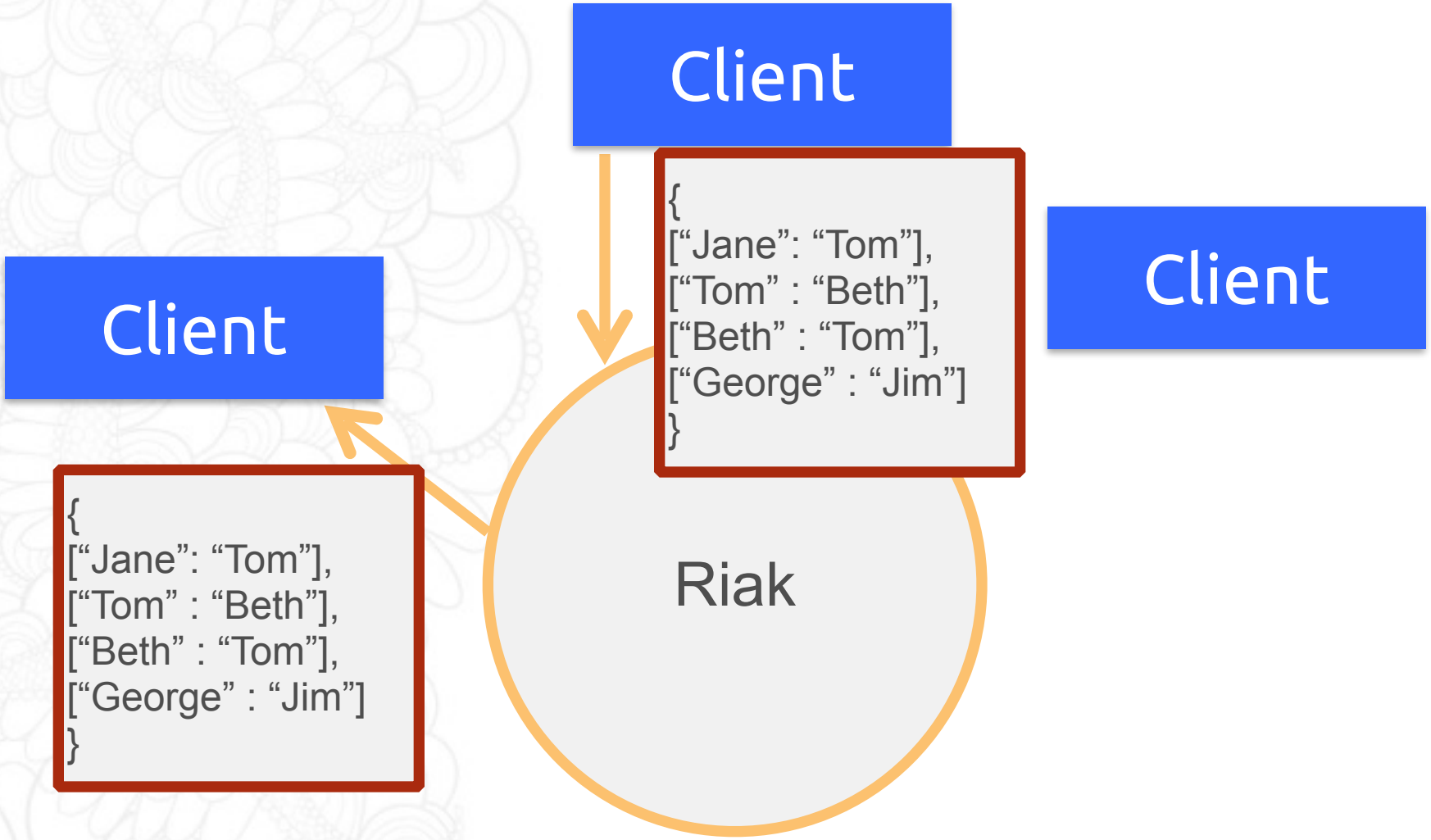
set conflict resolution



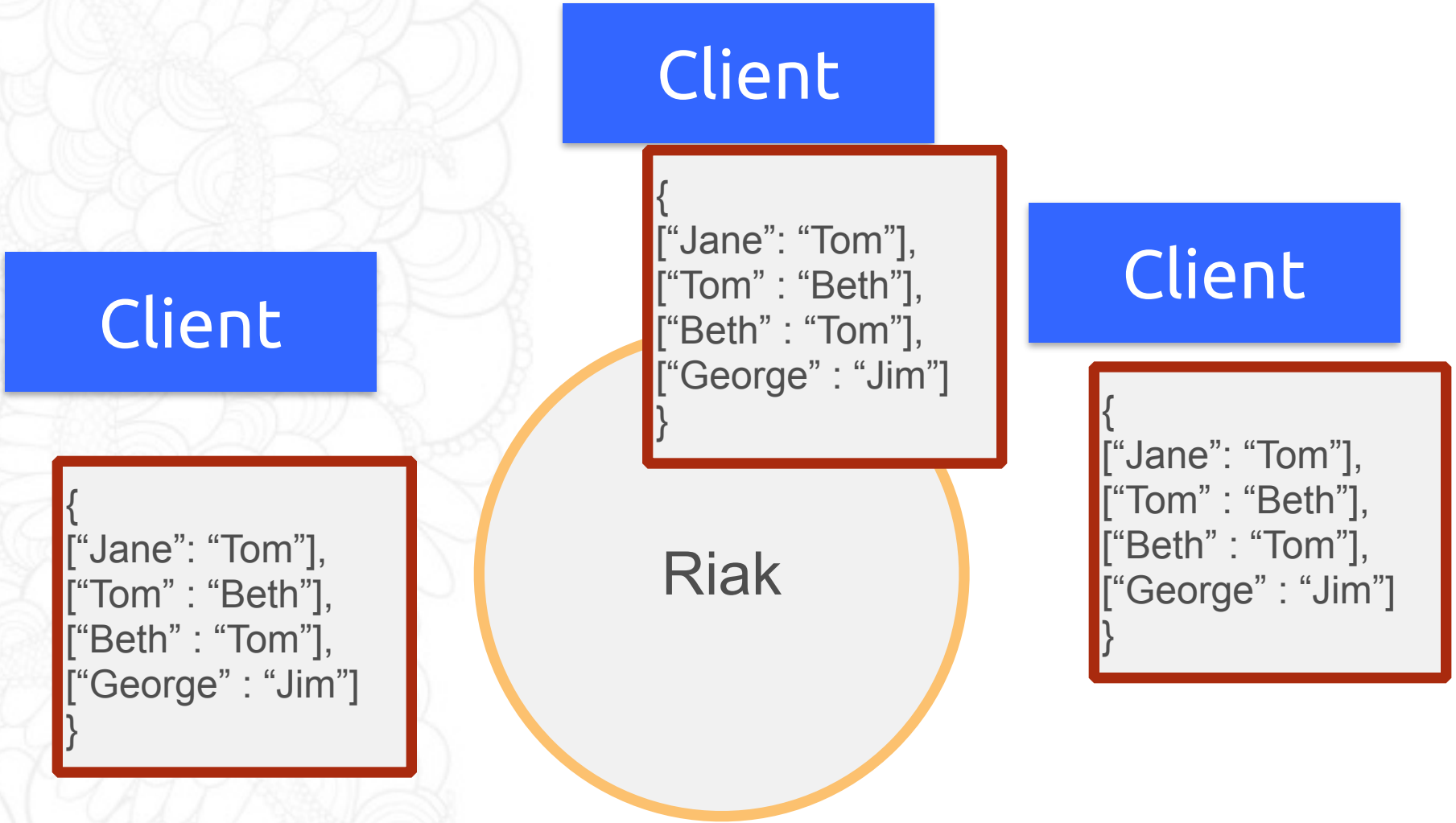
set conflict resolution



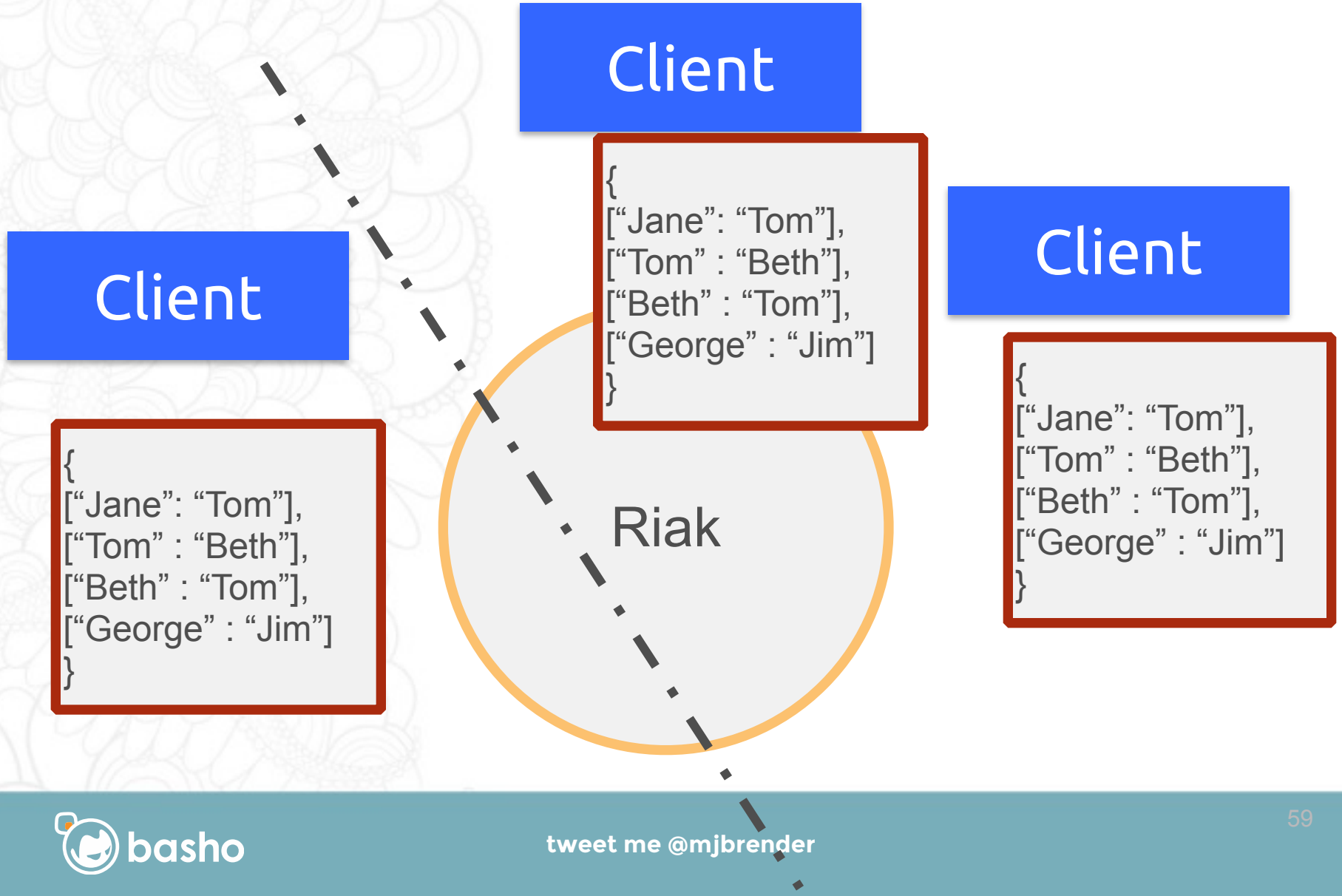
set conflict resolution



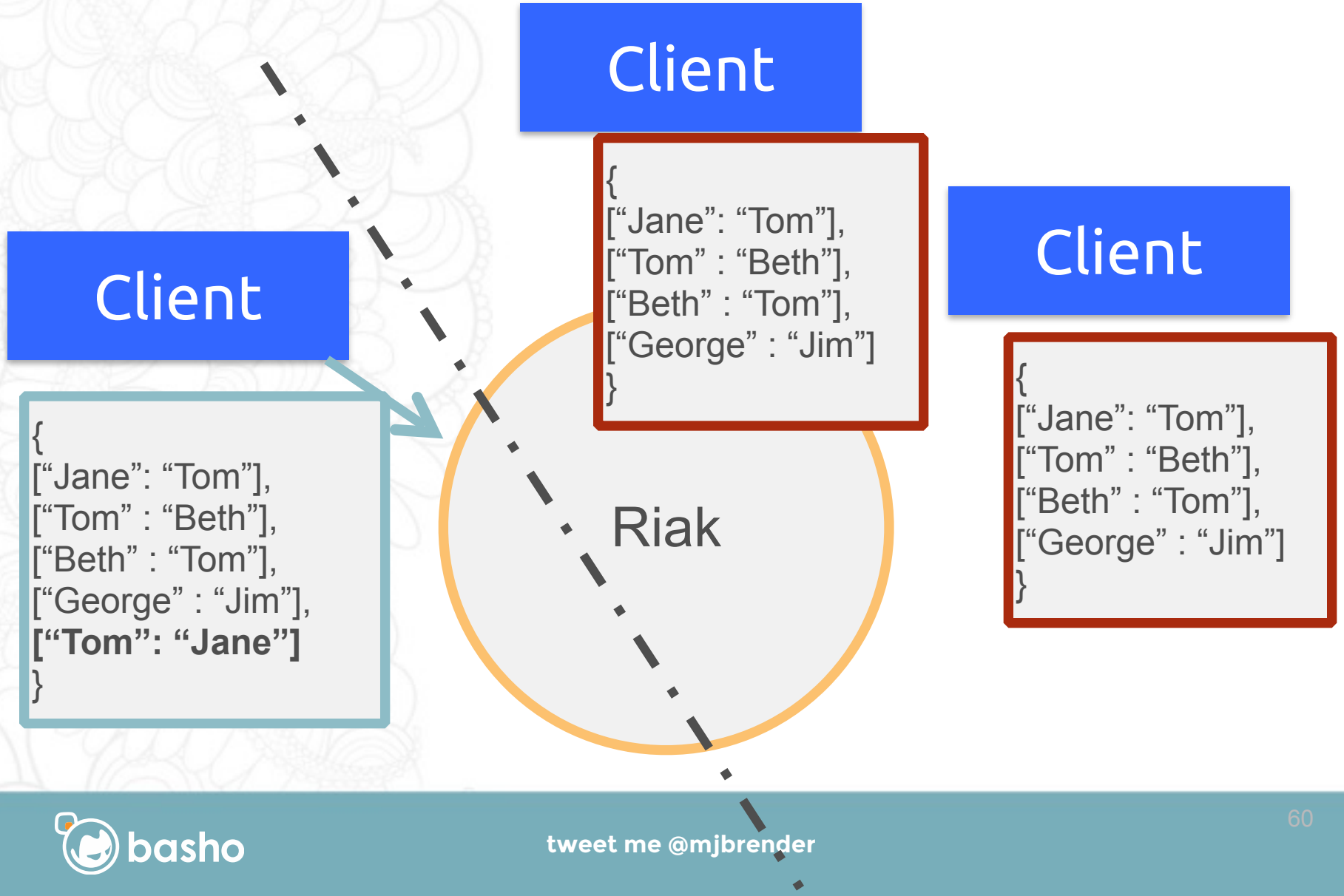
set conflict resolution



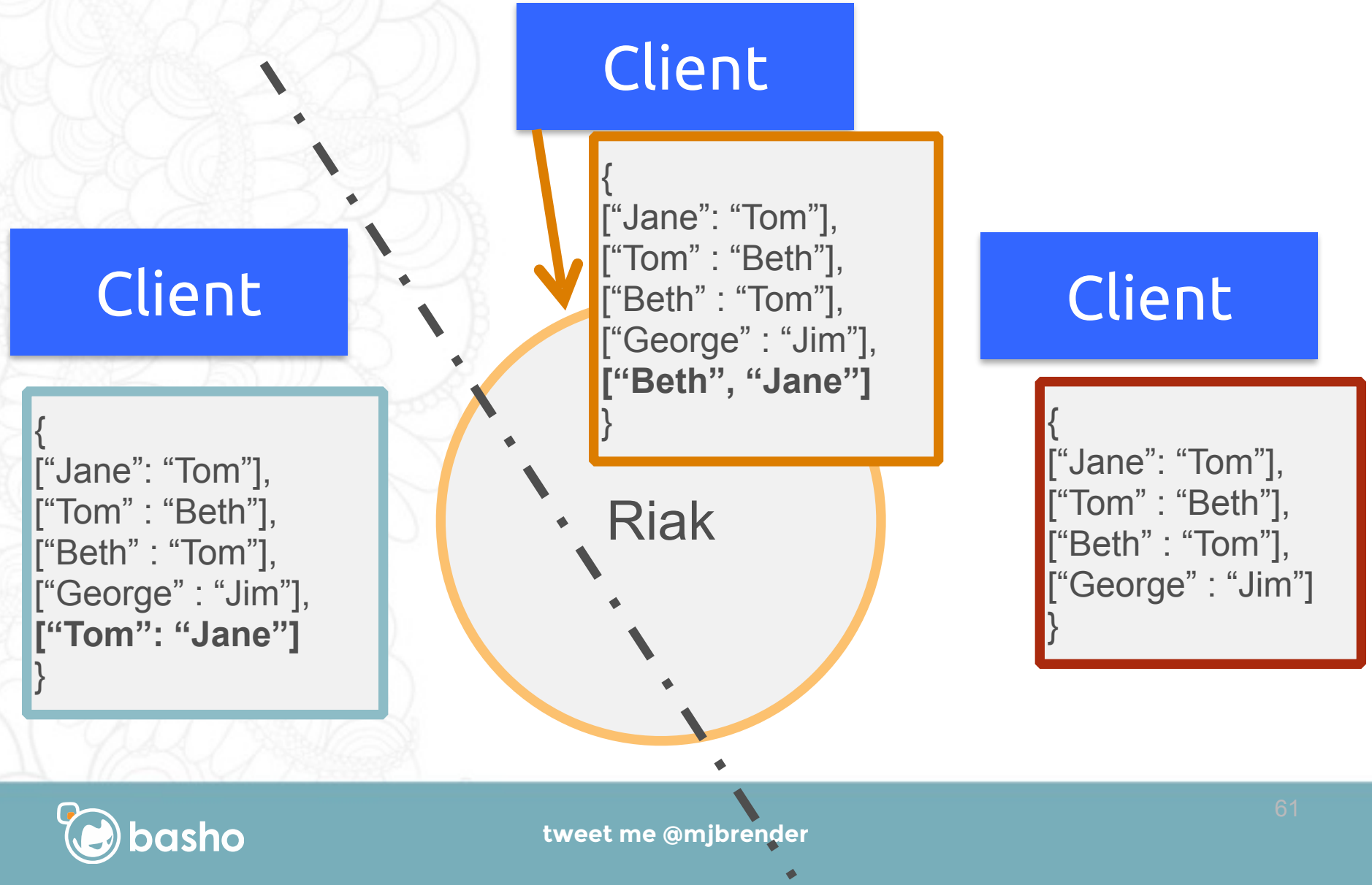
set conflict resolution



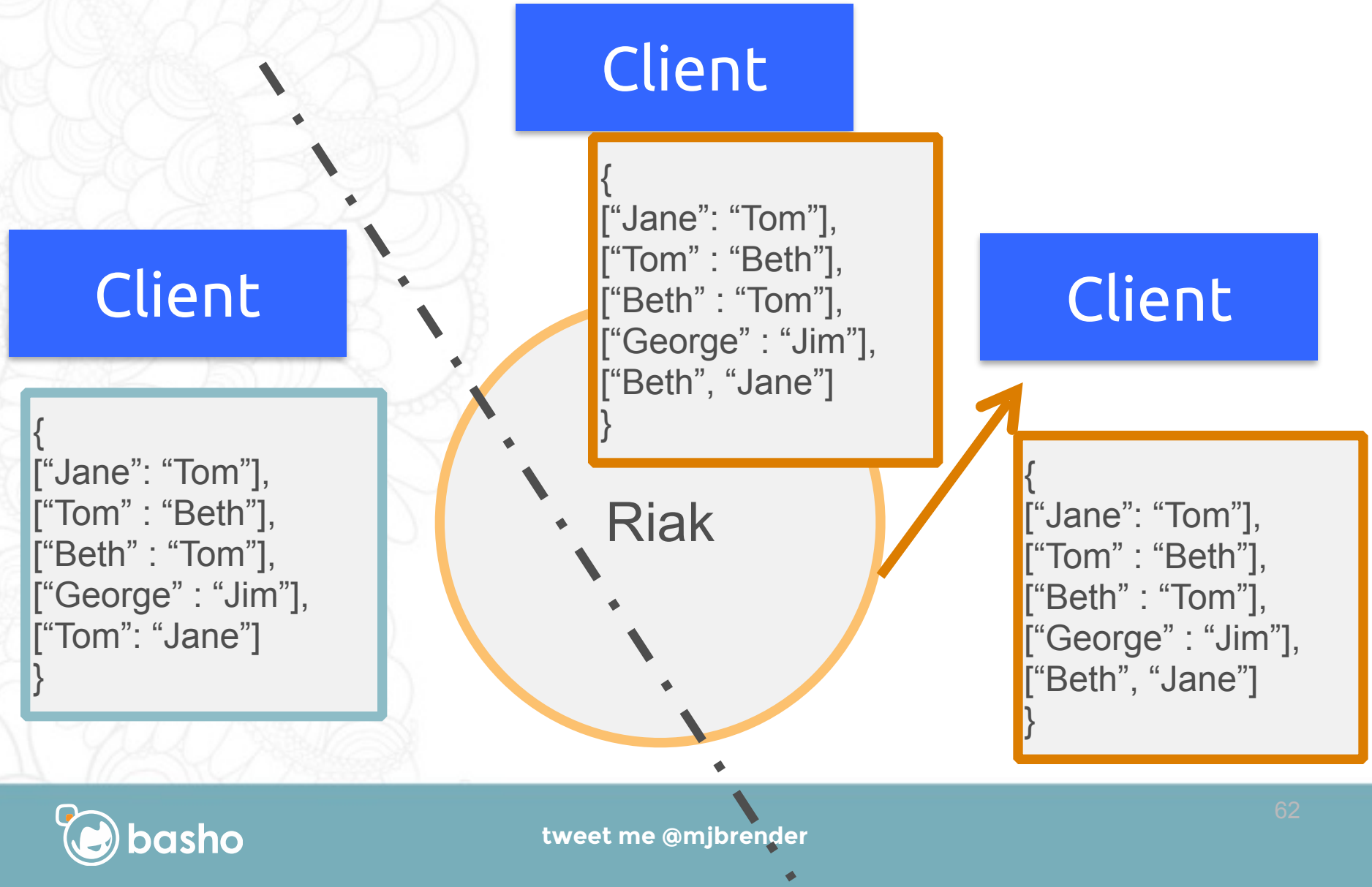
set conflict resolution



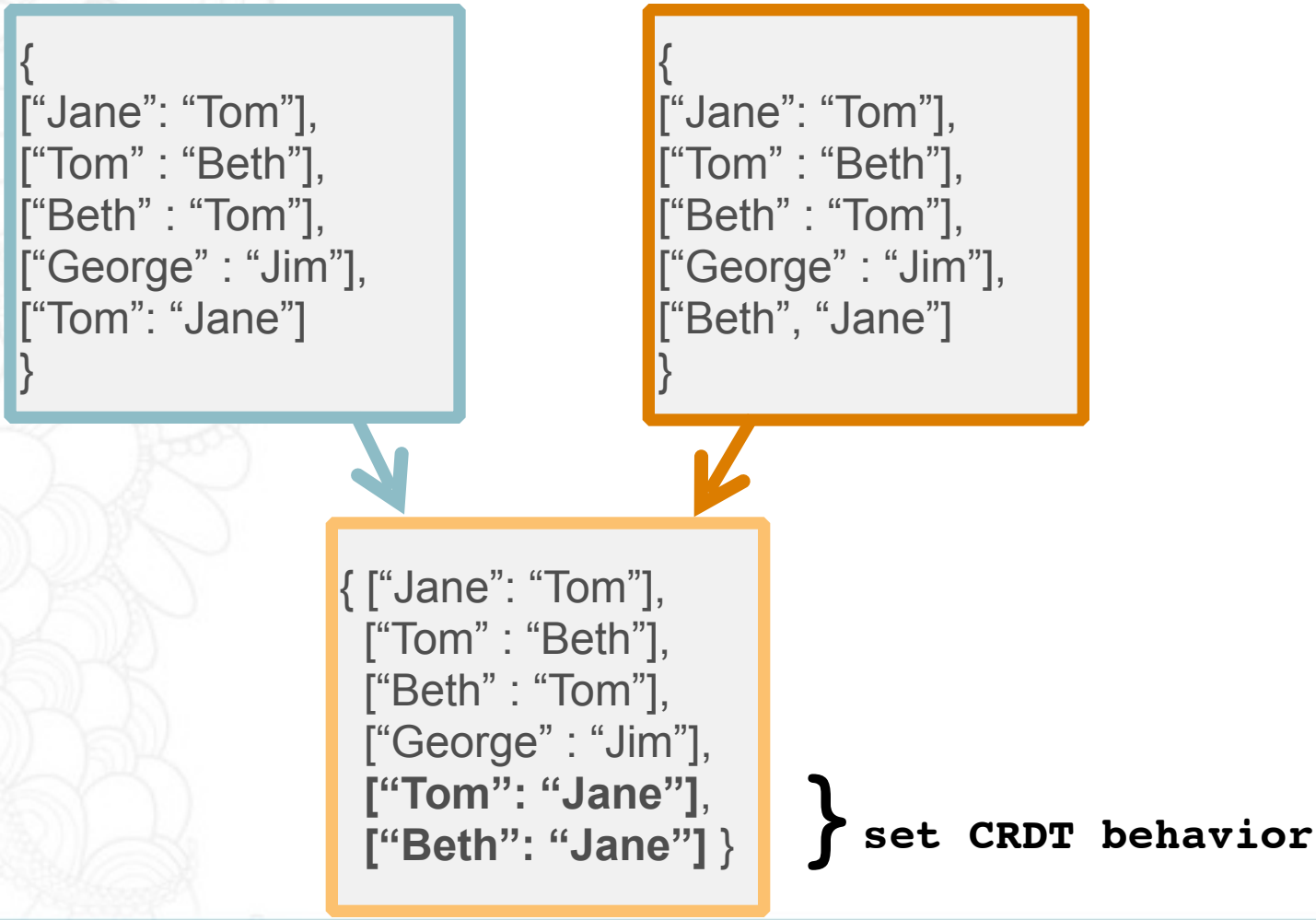
set conflict resolution



set conflict resolution



set conflict resolution



The Choices

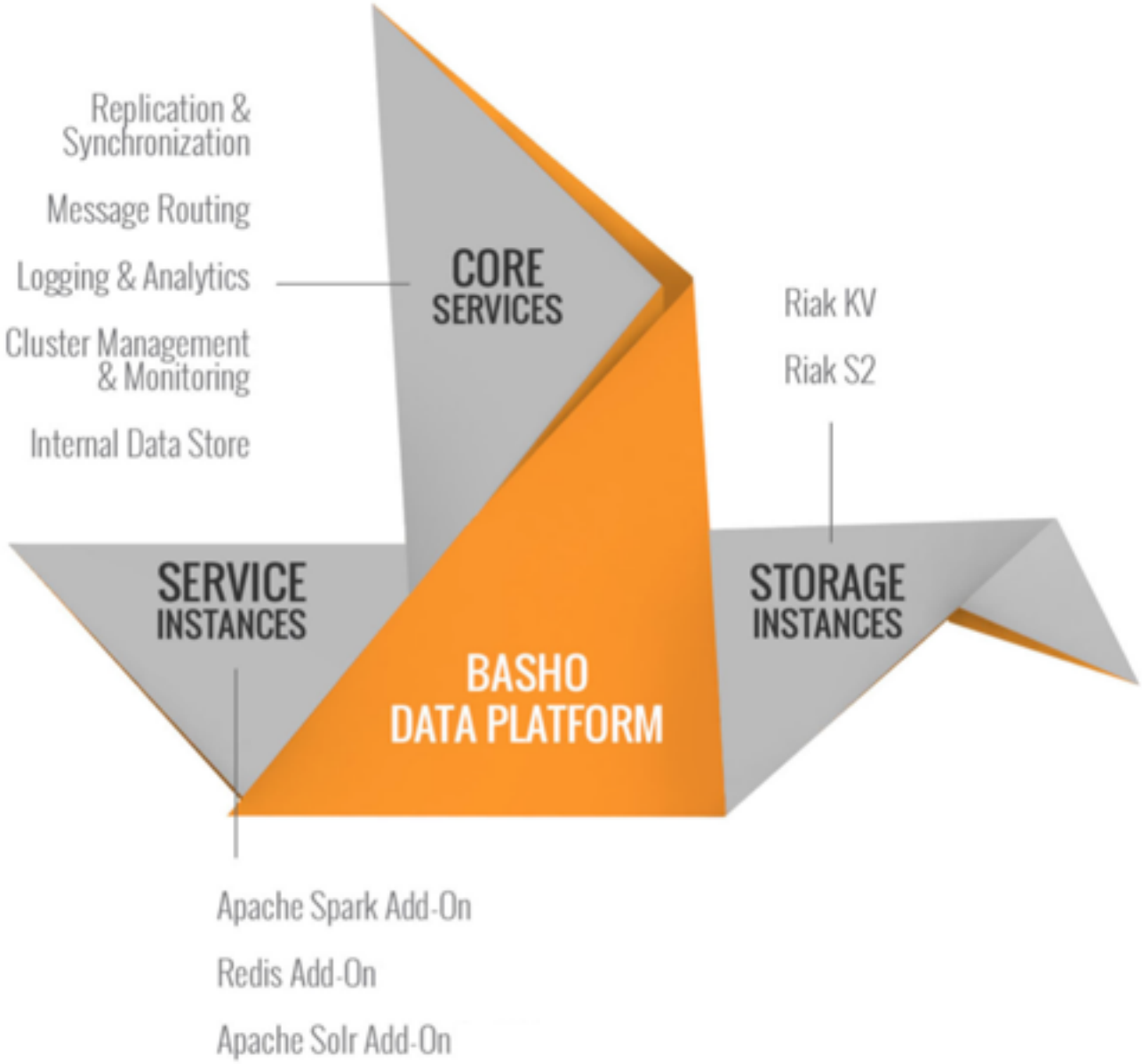
What Matters Most

What do I need (most) from my database?

- Consistency?
- Transactionality?
- Availability (and Performance)?

Where do I need most from my data?

- A single, scalable platform?
- A revolving door of new systems?
- A mix?


Replication & Synchronization
Message Routing
Logging & Analytics
Cluster Management & Monitoring
Internal Data Store
CORE SERVICES
Riak KV
Riak S2
SERVICE INSTANCES
STORAGE INSTANCES
BASHO DATA PLATFORM
Apache Spark Add-On
Redis Add-On
Apache Solr Add-On

Closing Questions

Do I *have* to scale?

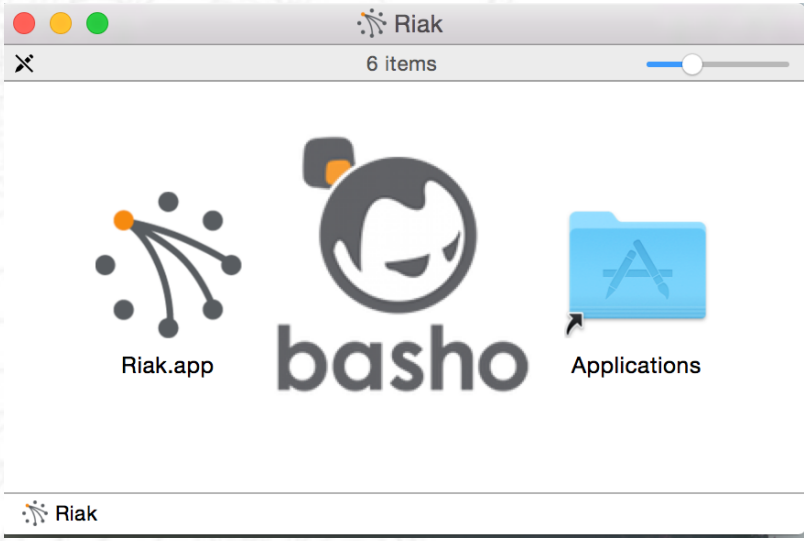
Will I *ever* want this?

Do we *have* to build it?

Get Hands-on

git clone

https://github.com/basho-labs



Thank You

Matt Brender
@mjbrender