RED HAT DEVELOPERS

Migrating to Microservice Databases:
From Relational Monolith to Distributed Data

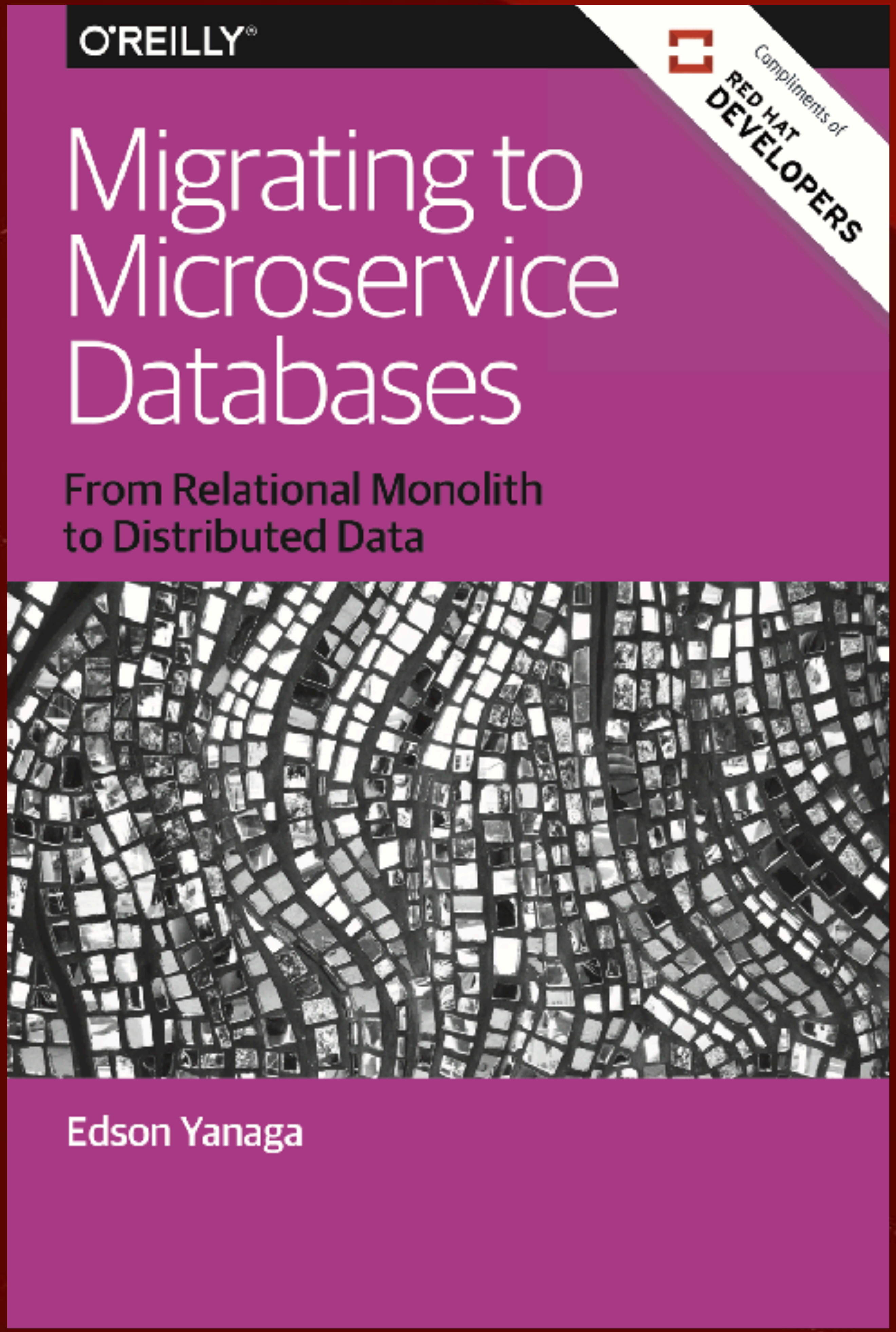
Edson Yanaga

Director of Developer Experience

@yanaga

Java Champion

Microsoft MVP



O'REILLY®

Compliments of RED HAT DEVELOPERS

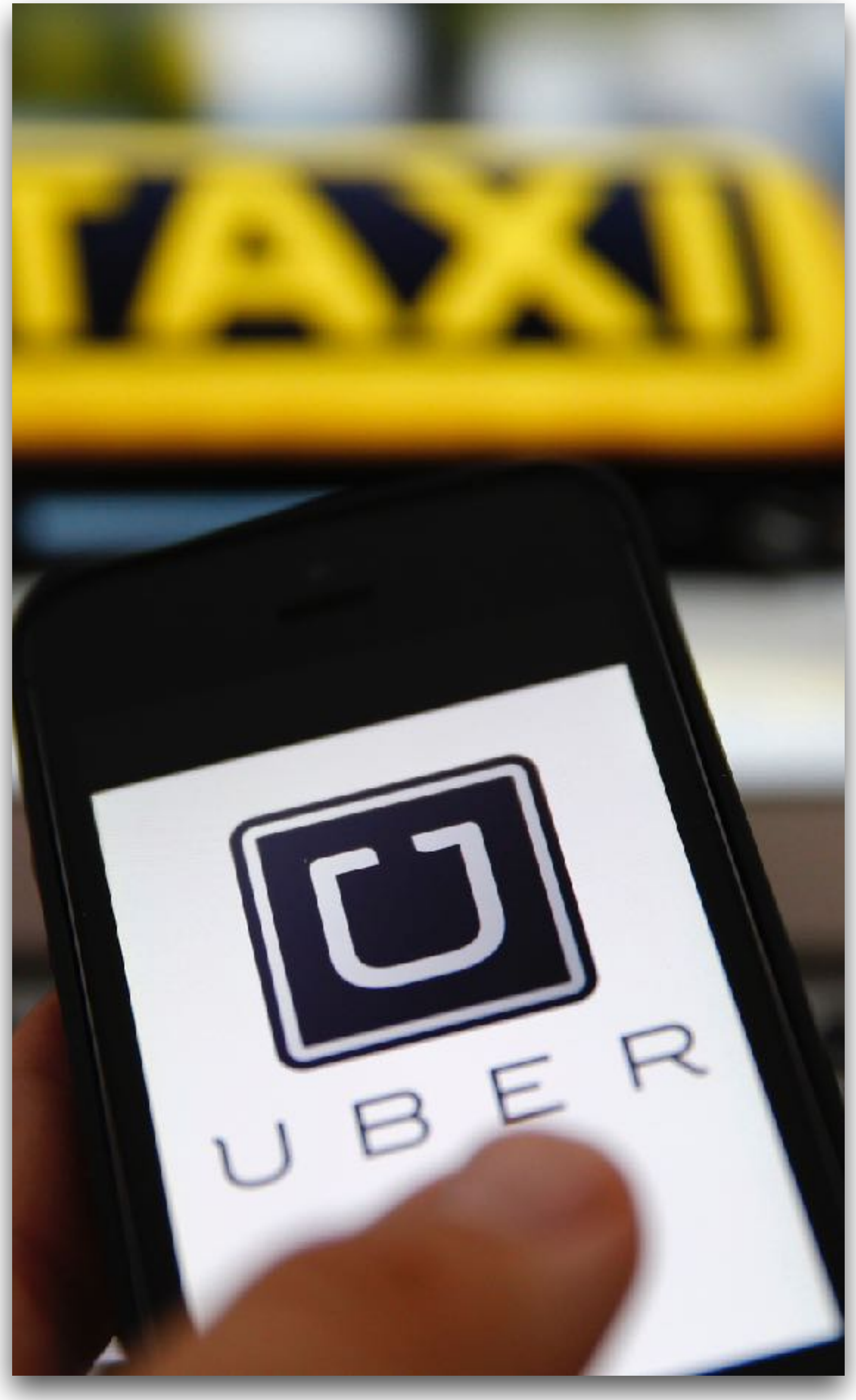
Migrating to Microservice Databases

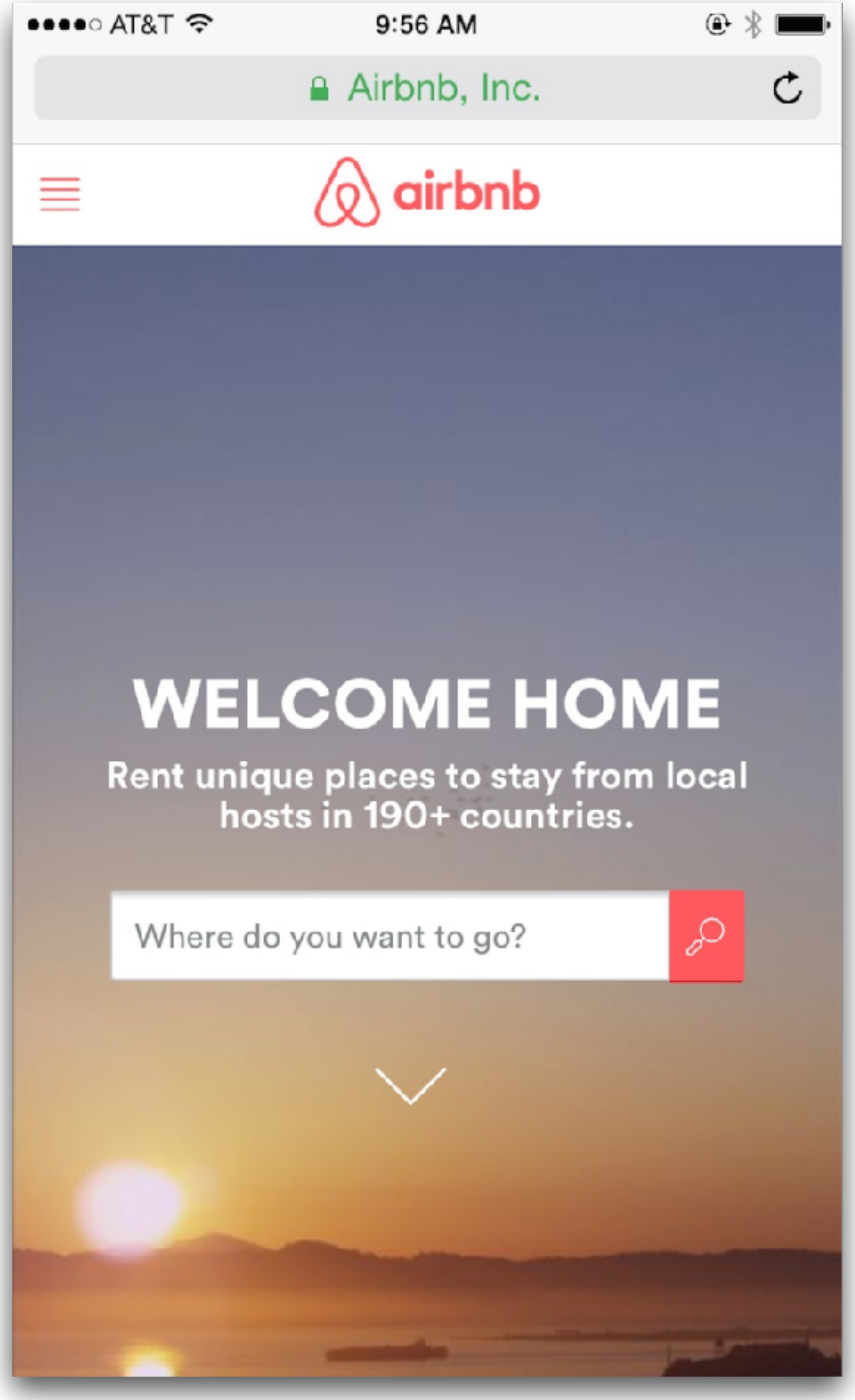
From Relational Monolith to Distributed Data

Edson Yanaga

http://developers.redhat.com/promotions/migrating-to-microservice-databases

“Now, every company is a software company”
— Forbes





WELCOME HOME

Rent unique places to stay from local hosts in 190+ countries.

Where do you want to go?

DevOps & Microservices

Feedback Loop

Batch Size

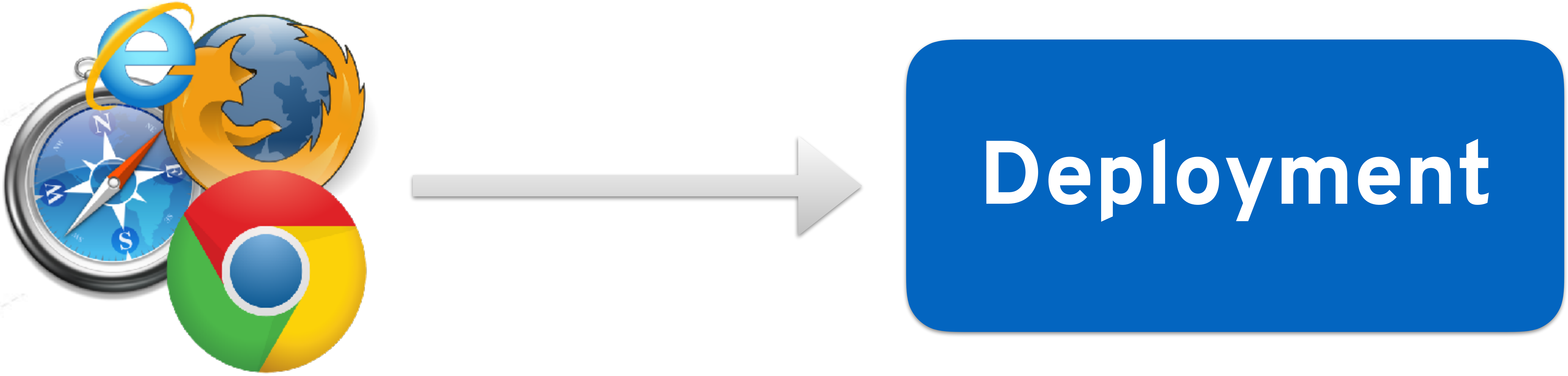
Maintenance Window

Zero Downtime

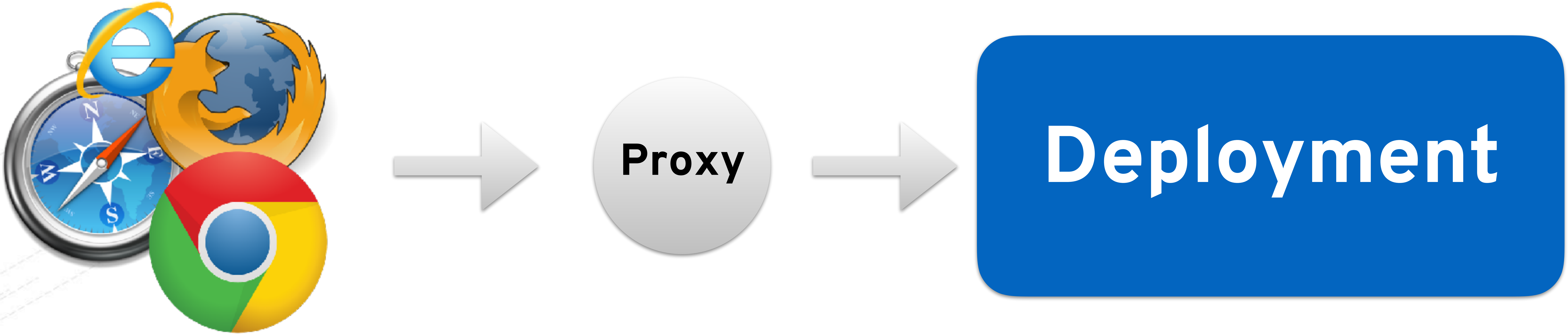
Blue

Green

Deployments

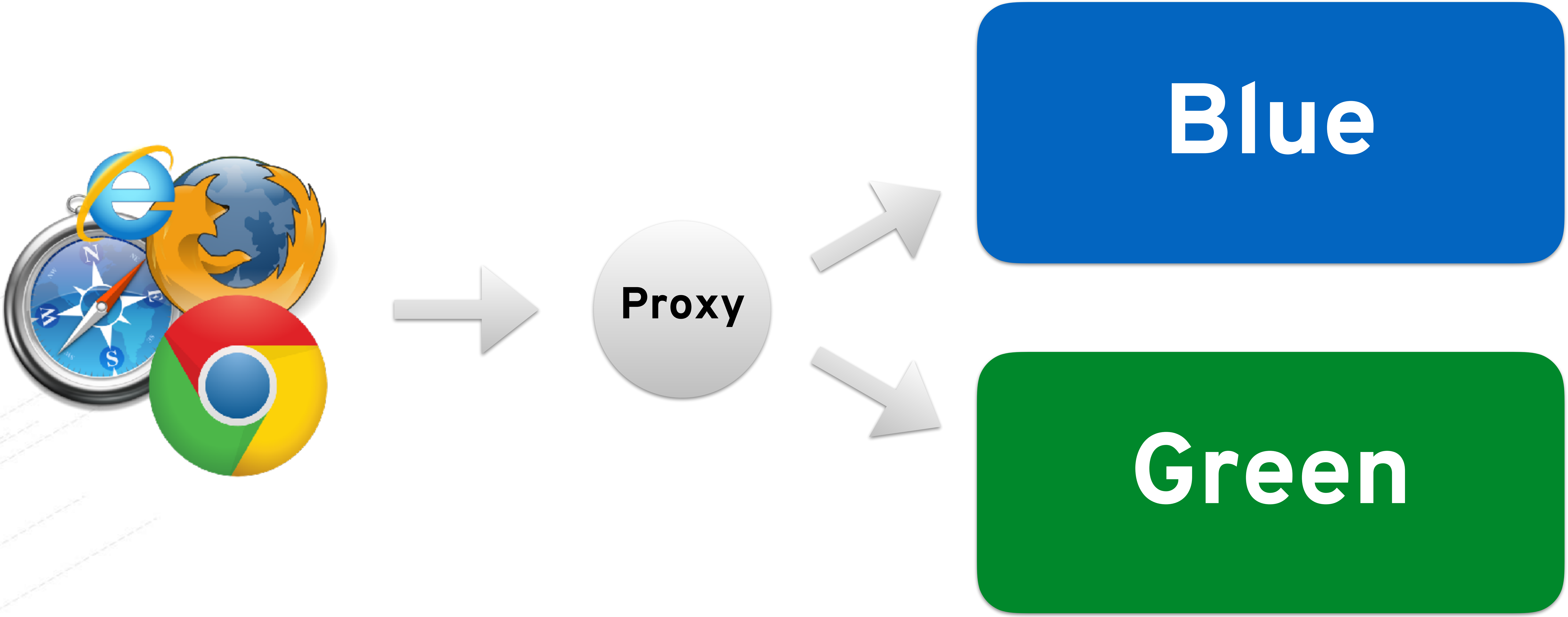


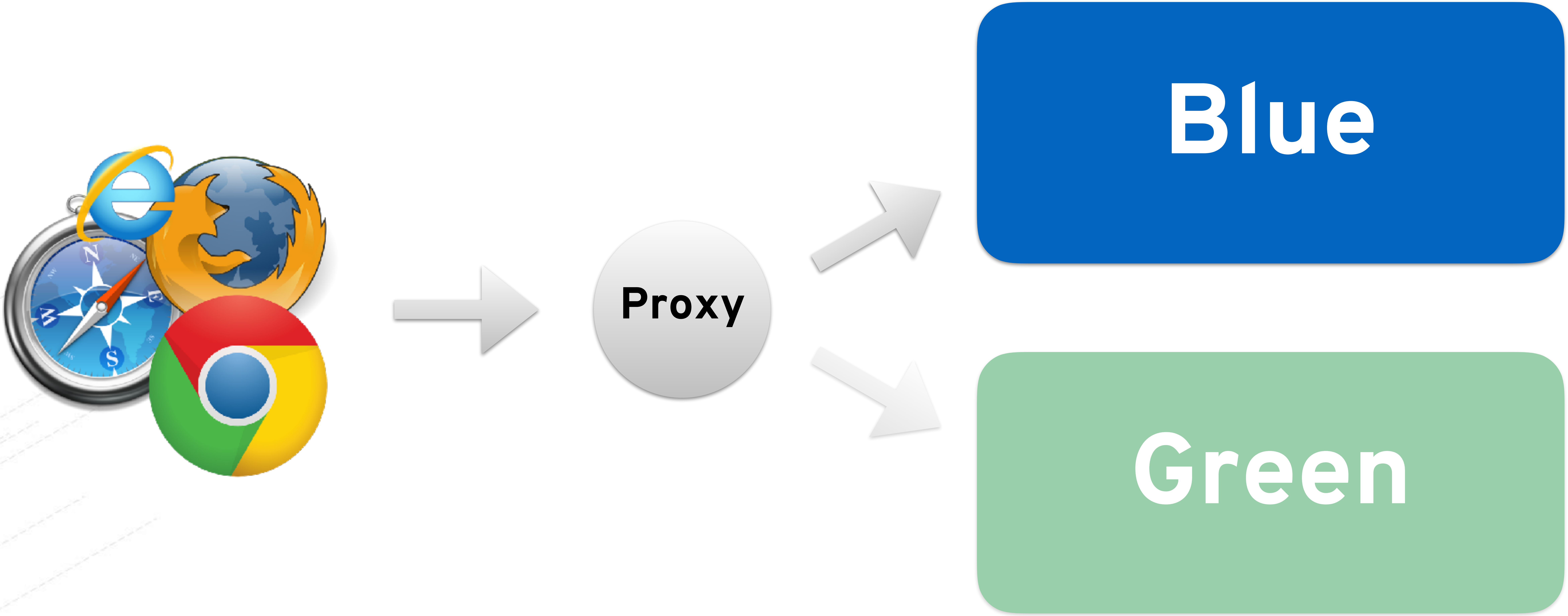
Deployment



Proxy

Deployment

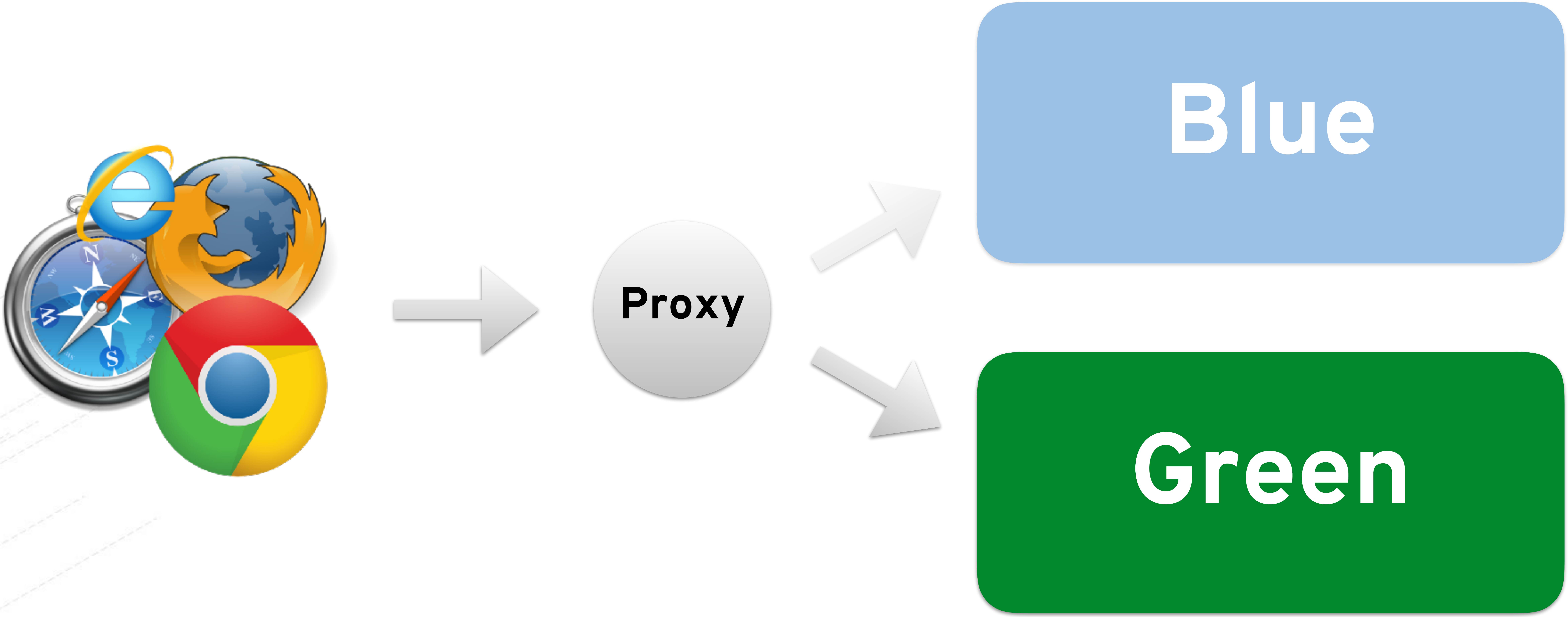

Proxy
Blue
Green



Proxy

Blue

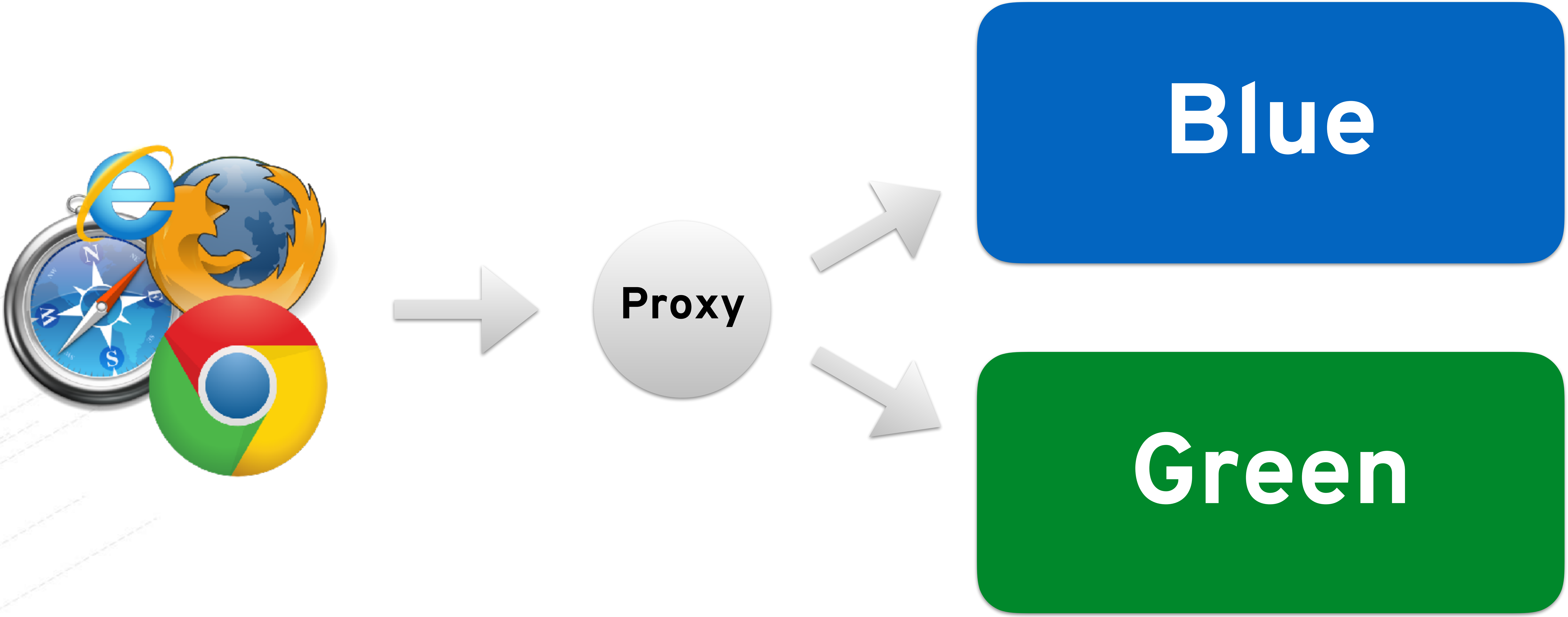
Green



Proxy

Blue

Green



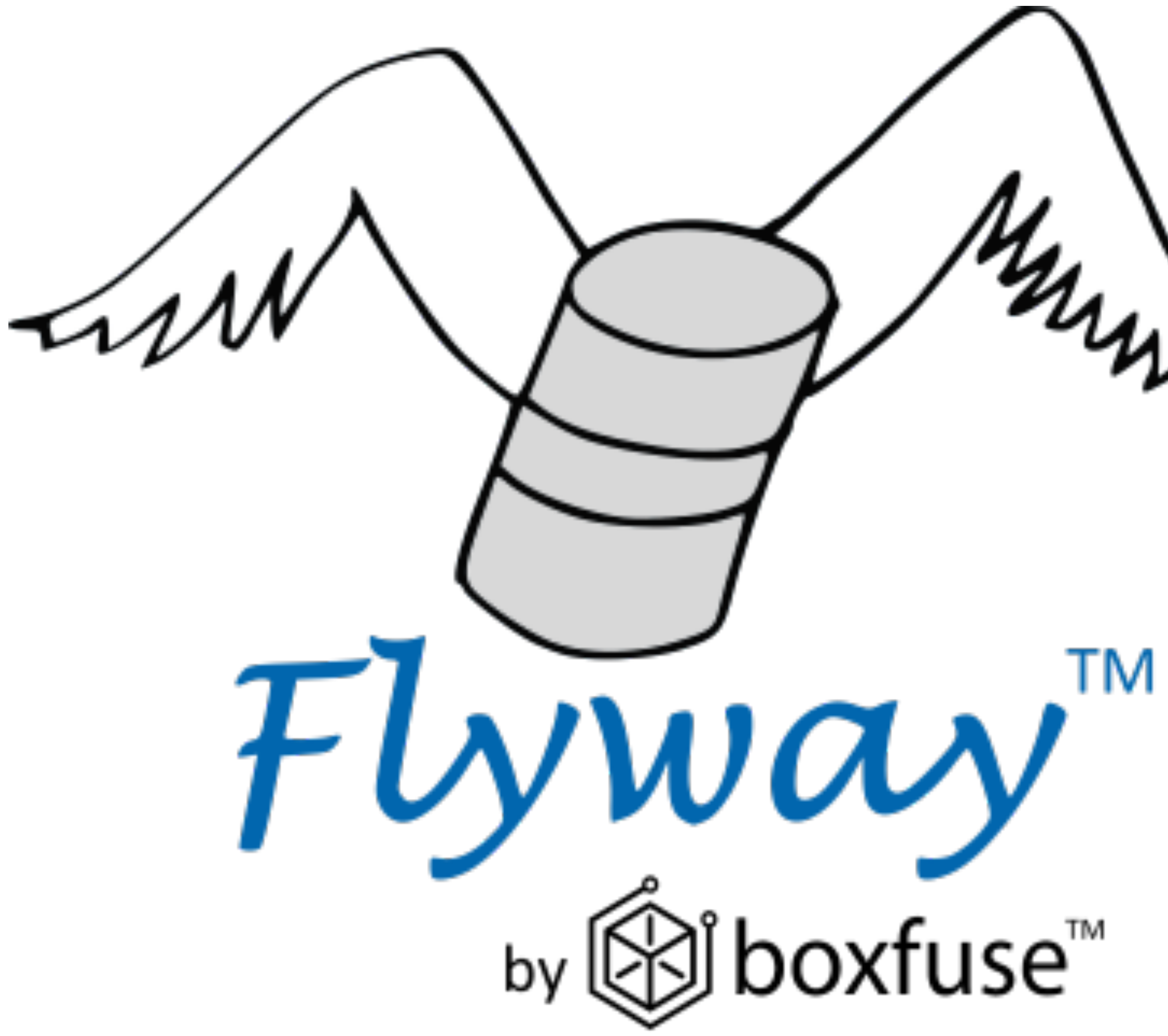
Proxy

Blue

Green

Code is easy, state is hard

What about my relational database?


Flyway™
by boxfuse™

LIQUIBASE

Zero Downtime Migrations

Back and Forward Compatibility

Baby Steps = Smallest Possible Batch Size

Too many rows = Long Locks

Shard your updates

```sql
ALTER TABLE customers RENAME COLUMN wrong TO correct;
ALTER TABLE customers ADD COLUMN correct VARCHAR(20);

UPDATE customers SET correct = wrong
  WHERE id BETWEEN 1 AND 100;

UPDATE customers SET correct = wrong
  WHERE id BETWEEN 101 AND 200;

ALTER TABLE customers DELETE COLUMN wrong;
```

Scenarios

Add a Column

Rename a Column

Change Type/Format of a Column

Delete a Column

Add a Column

1 ADD COLUMN

2 Code computes the read value and writes to new column

3 Update data using shards

4 Code reads and writes from the new column

Rename a Column

1 ADD COLUMN

2 Code reads from the old column and writes to both

3 Copy data using small shards

4 Code reads from the new column and writes to both

5 Code reads and writes from the new column

6 Delete the old column (later)

Change Type/Format of a Column

1 ADD COLUMN

2 Code reads from the old column and writes to both

3 Copy data using small shards

4 Code reads from the new column and writes to both

5 Code reads and writes from the new column

6 Delete the old column (later)

Delete a Column

1 DON’T

2 Stop using the read value but keep writing to the column

3 Delete the column

What about referential integrity constraints?

RED HAT
DEVELOPERS

Drop them and recreate when migration is done

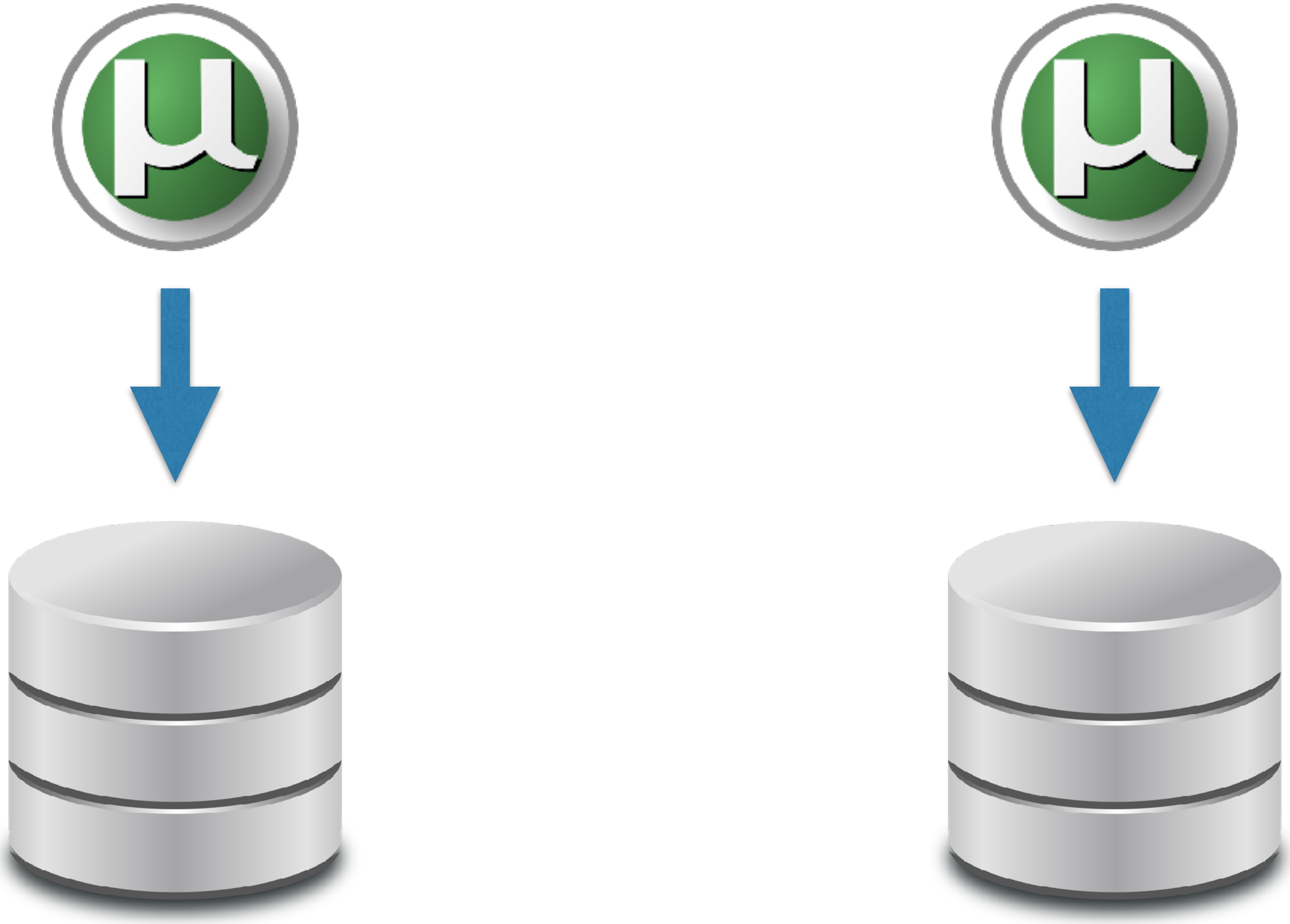
Microservices Characteristics

https://martinfowler.com/microservices/

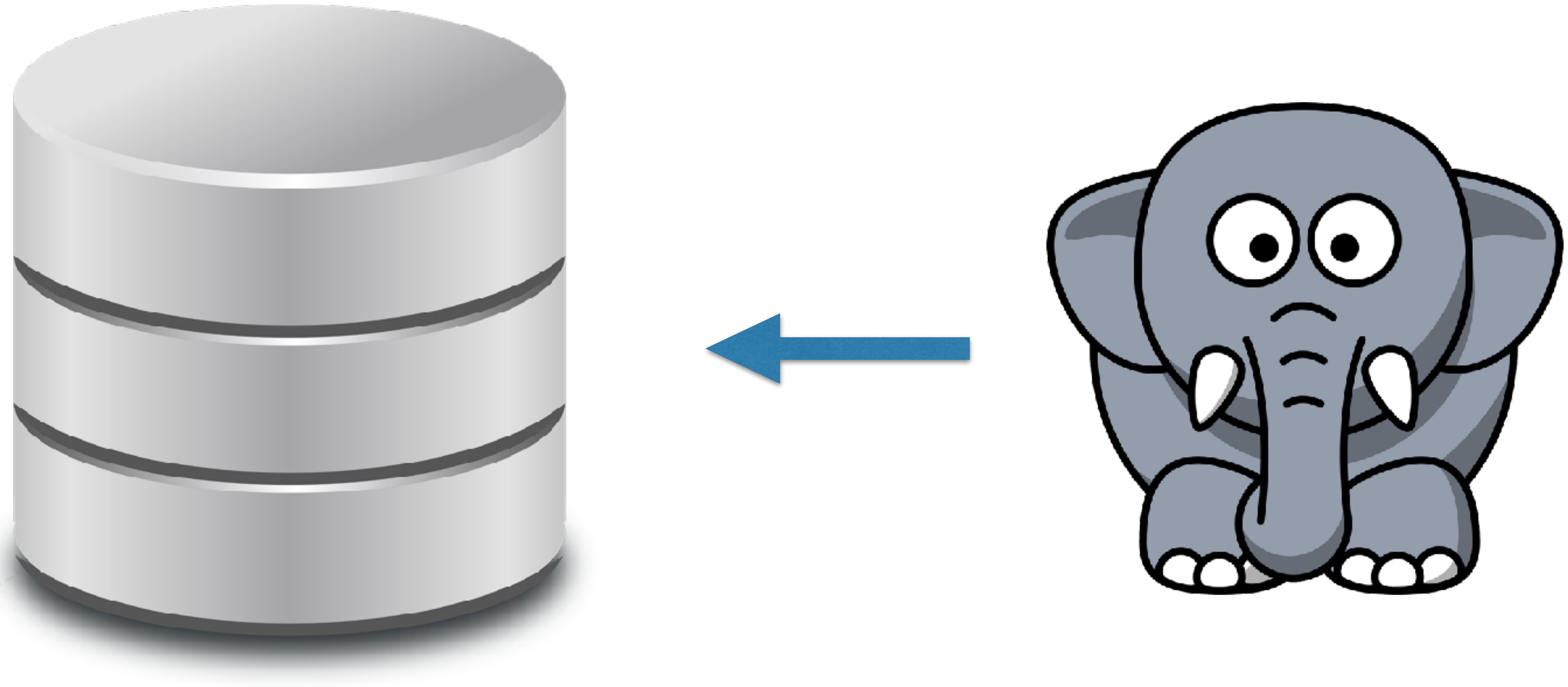
- Componentization via Services
- Organized around Business Capabilities
- Products not Projects
- Smart endpoints and dumb pipes
- Decentralized Governance
- **Decentralized Data Management**
- Infrastructure Automation
- Design for failure
- Evolutionary Design

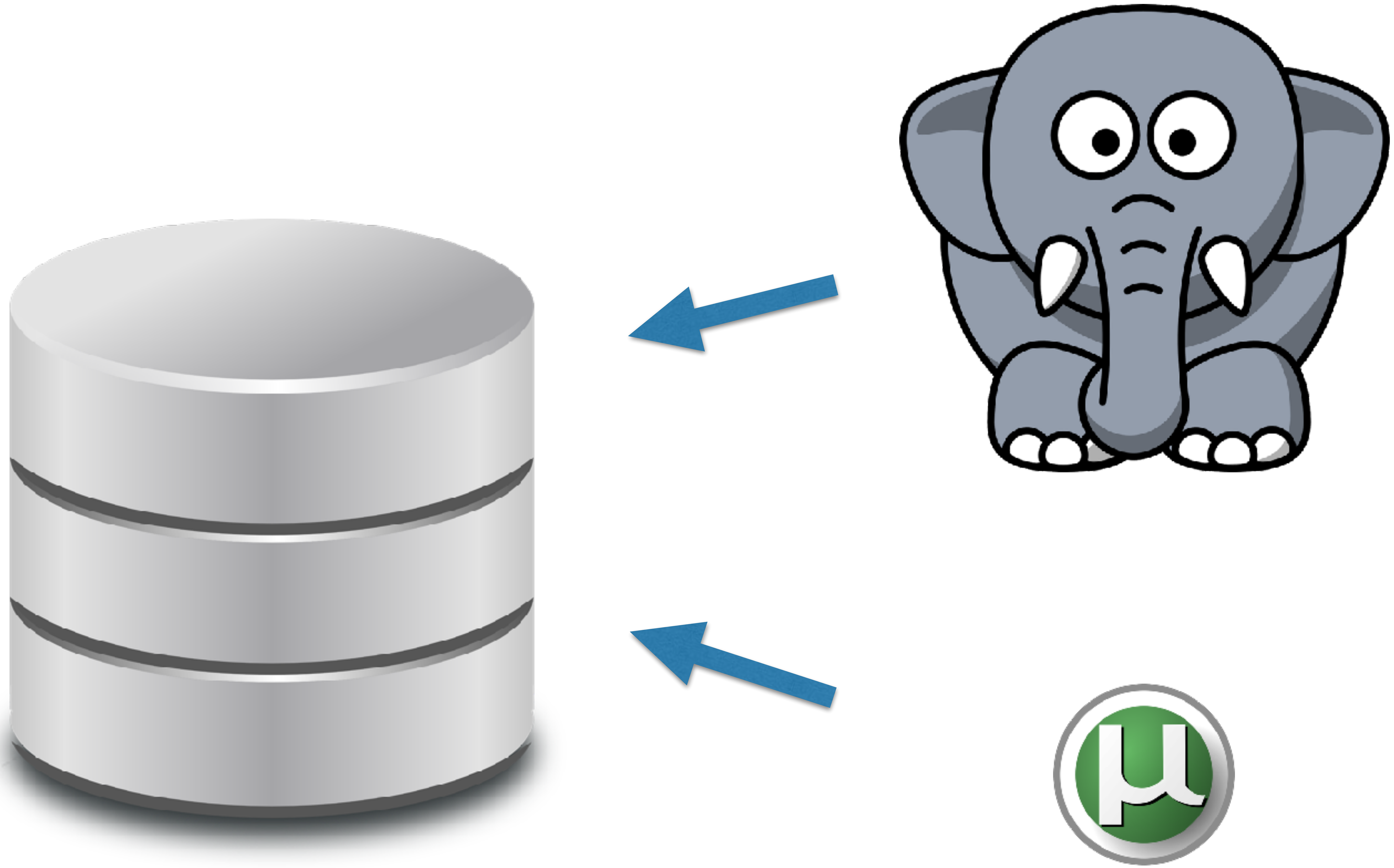
Extracting your Microservice database

One database per Microservice



But I have a monolithic database!





Splitting is not easy, but how do I integrate later?

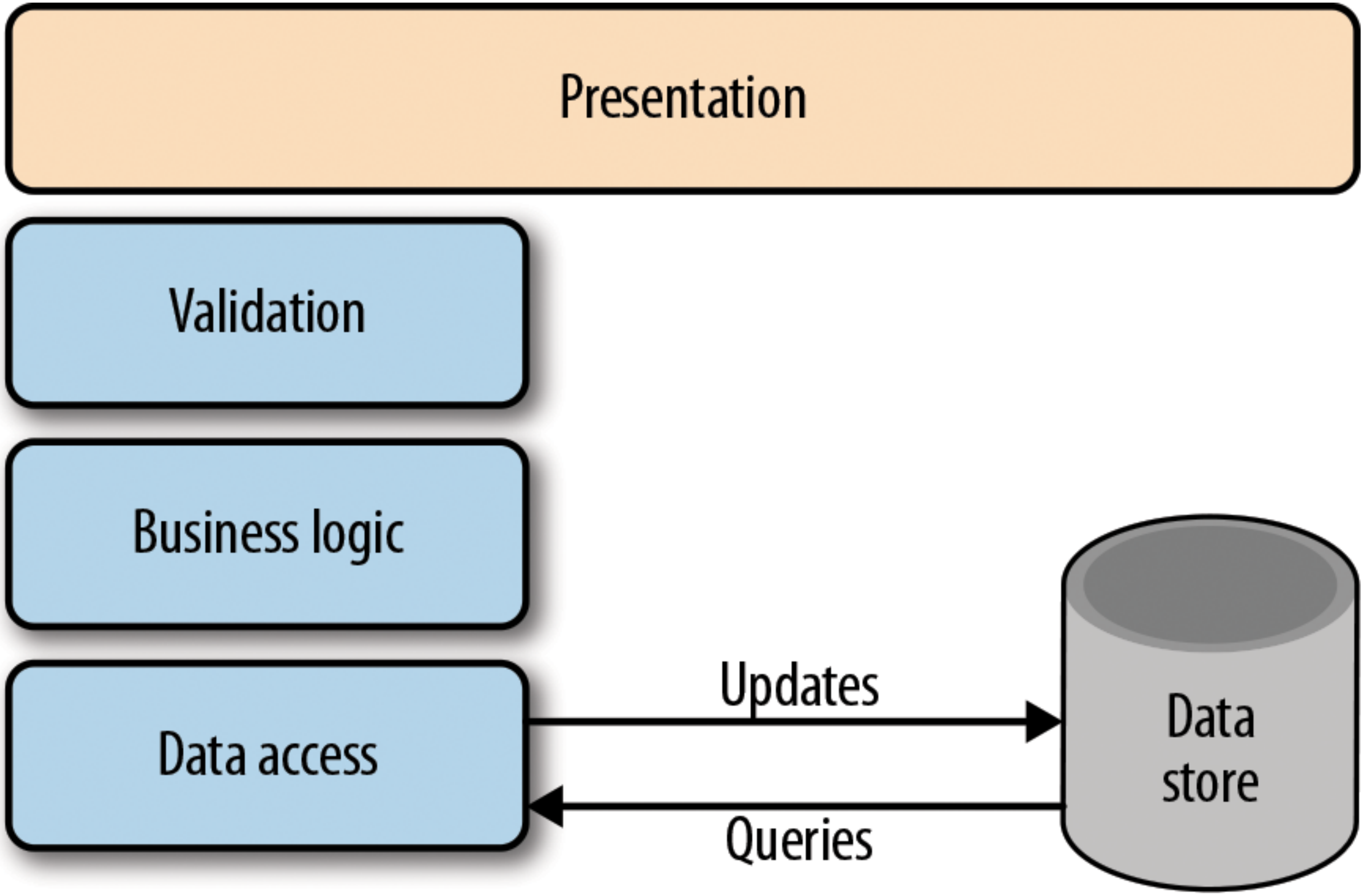
Consistency Models

Strong Consistency

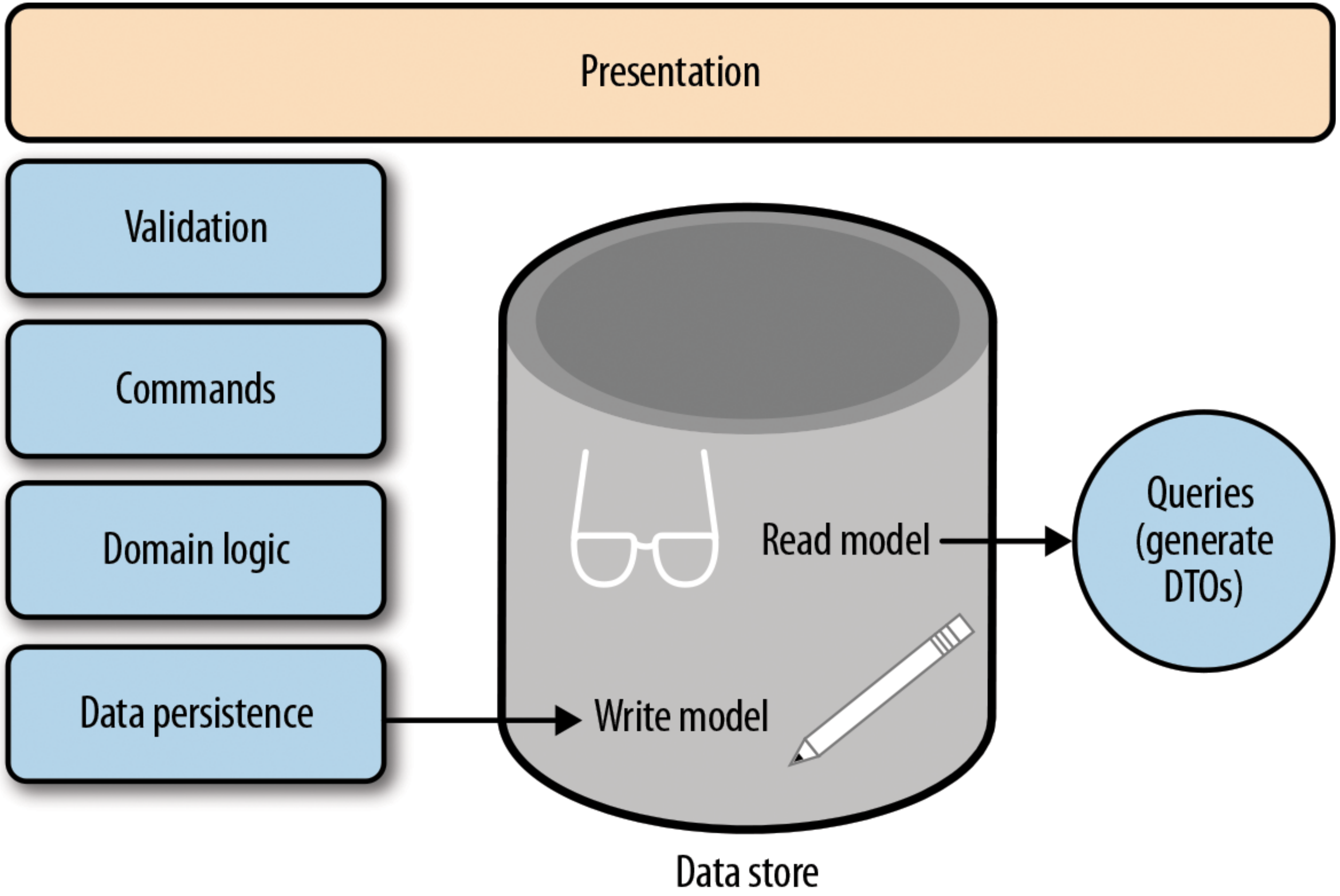
Eventual Consistency

CRUD & CQRS

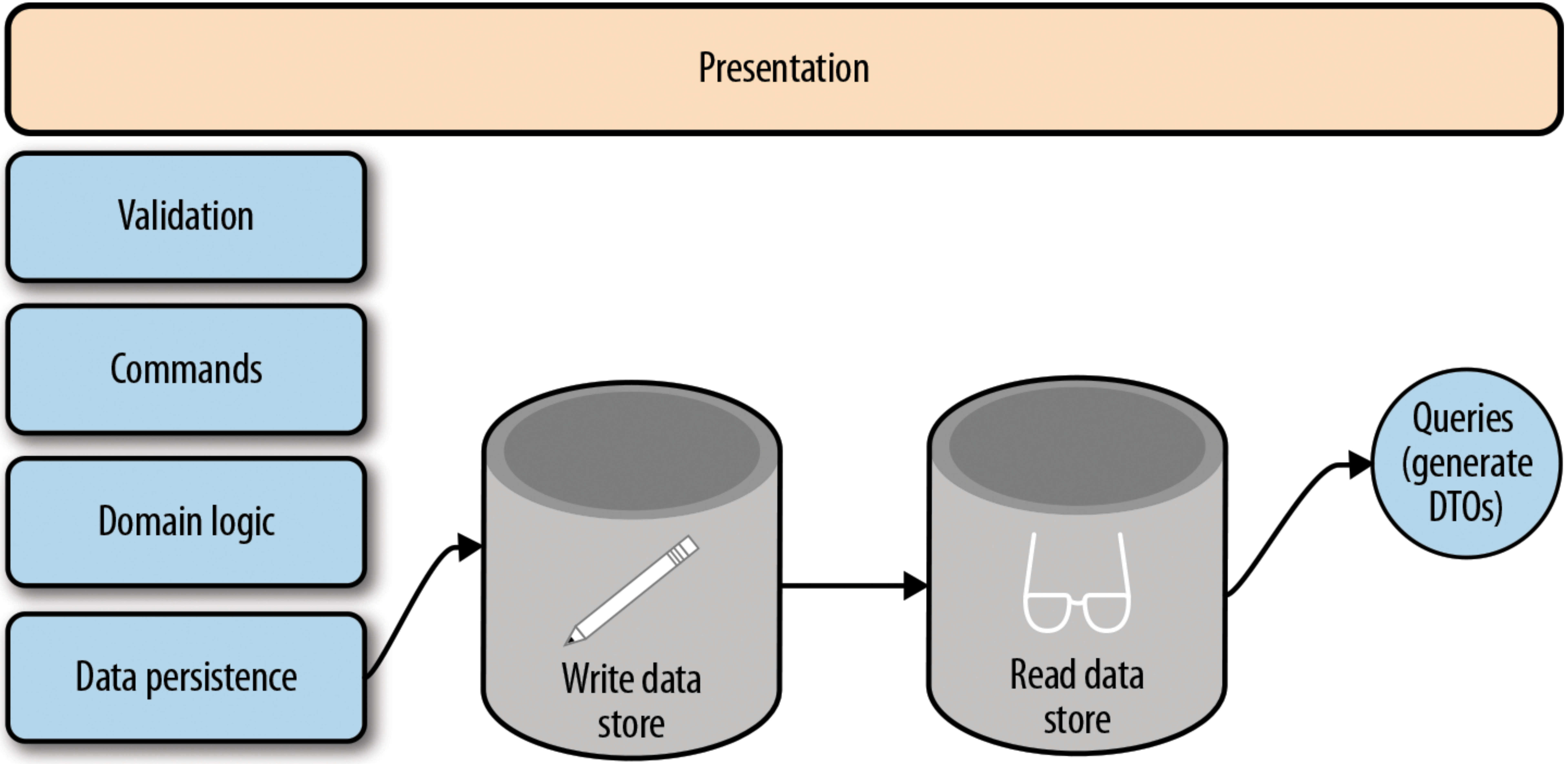
CRUD (Create, Read, Update, Delete)



CQRS (Command Query Responsibility Segregation)



CQRS with separate data stores



CQRS & Event Sourcing

Scenarios

Shared Tables

Database View

Database Materialized View

Mirror Table using Trigger

Mirror Table using Transactional Code

Mirror Table using ETL

Mirror Table using Data Virtualization

Event Sourcing

Change Data Capture

Shared Tables

Fastest Data Integration

Strong Consistency

Low cohesion and high coupling

Database View

Easiest one to implement

Largest support from DBMS vendors

Possible performance issues

Strong Consistency

One database must be reachable by the other

Updatable depending on DBMS support

Database Materialized View

Better performance

Strong or Eventual Consistency

One database must be reachable by the other

Updatable depending on DBMS support

Database Trigger

Depends on DBMS Support

Strong Consistency

One database must be reachable by the other

Transactional Code

Any code: usually Stored Procedures or Distributed Transactions

Strong Consistency

Possible cohesion/coupling issues

Possible performance issues

Updatable depending on how it is implemented

ETL Tools

Lots of available tools

Requires external trigger (usually time-based)

Can aggregate from multiple datasources

Eventual Consistency

Read only integration

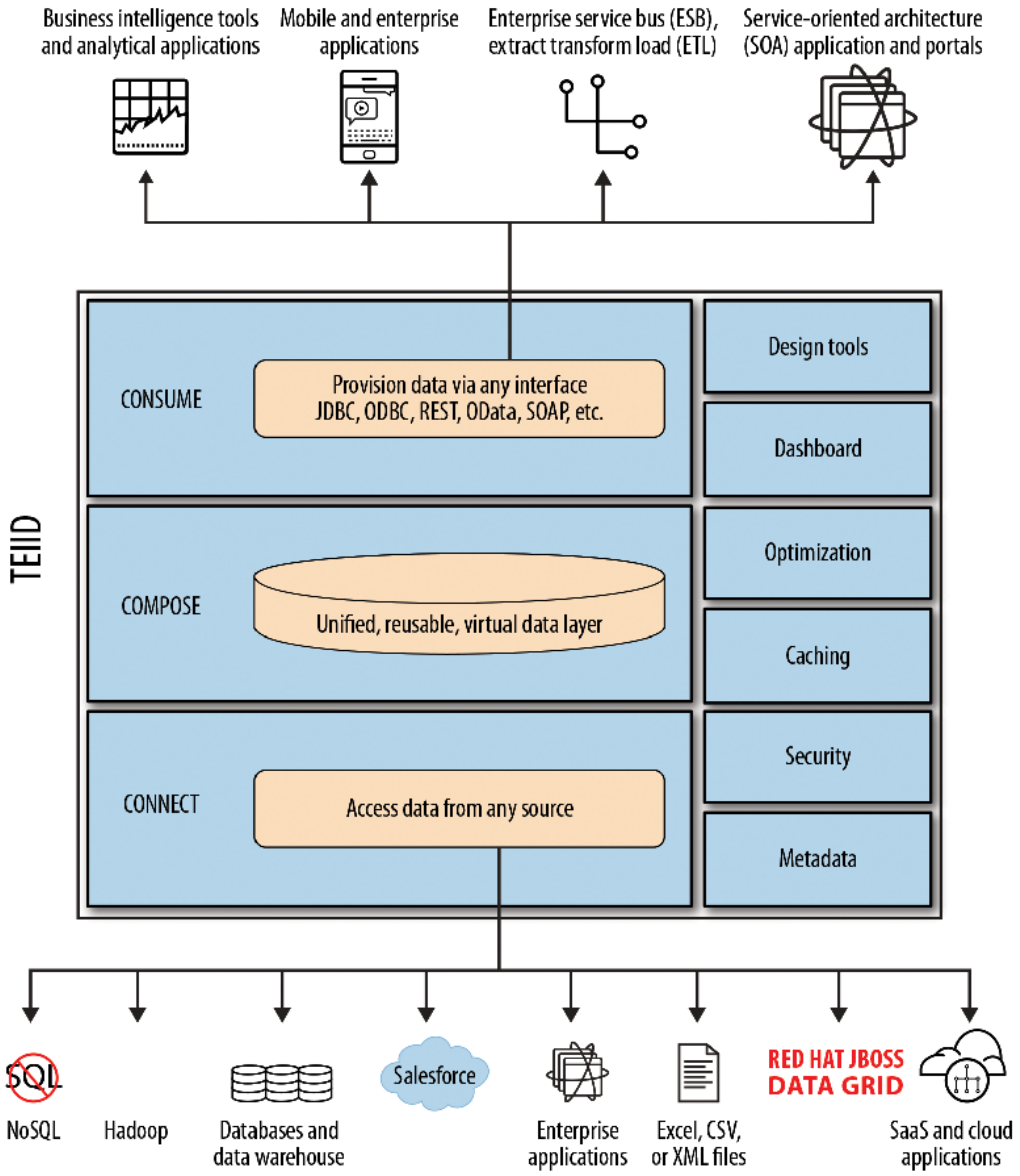
Data Virtualization

Real Time Access

Strong Consistency

Can aggregate from multiple datasources

Updatable depending on Data Virtualization Platform


Business intelligence tools and analytical applications
Mobile and enterprise applications
Enterprise service bus (ESB), extract transform load (ETL)
Service-oriented architecture (SOA) application and portals
TEIID
CONSUME
Provision data via any interface JDBC, ODBC, REST, OData, SOAP, etc.
COMPOSE
Unified, reusable, virtual data layer
CONNECT
Access data from any source
Design tools
Dashboard
Optimization
Caching
Security
Metadata
SQL
NoSQL
Hadoop
Databases and data warehouse
Salesforce
Enterprise applications
Excel, CSV, or XML files
RED HAT JBOSS DATA GRID
SaaS and cloud applications

Data Sources

Event Sourcing

State of data is a stream of events

Eases auditing

Eventual Consistency

Usually combined with a Message Bus

High scalability

Change Data Capture

Read datasource is updated through a stream of events

Eventual Consistency

Usually combined with a Message Bus

High scalability

http://debezium.io

Join

developers.redhat.com

Feedback welcome!

@yanaga

RED HAT DEVELOPERS

Thank you!

- plus.google.com/+RedHat
- facebook.com/redhatinc
- linkedin.com/company/red-hat
- twitter.com/RedHatNews
- youtube.com/user/RedHatVideos