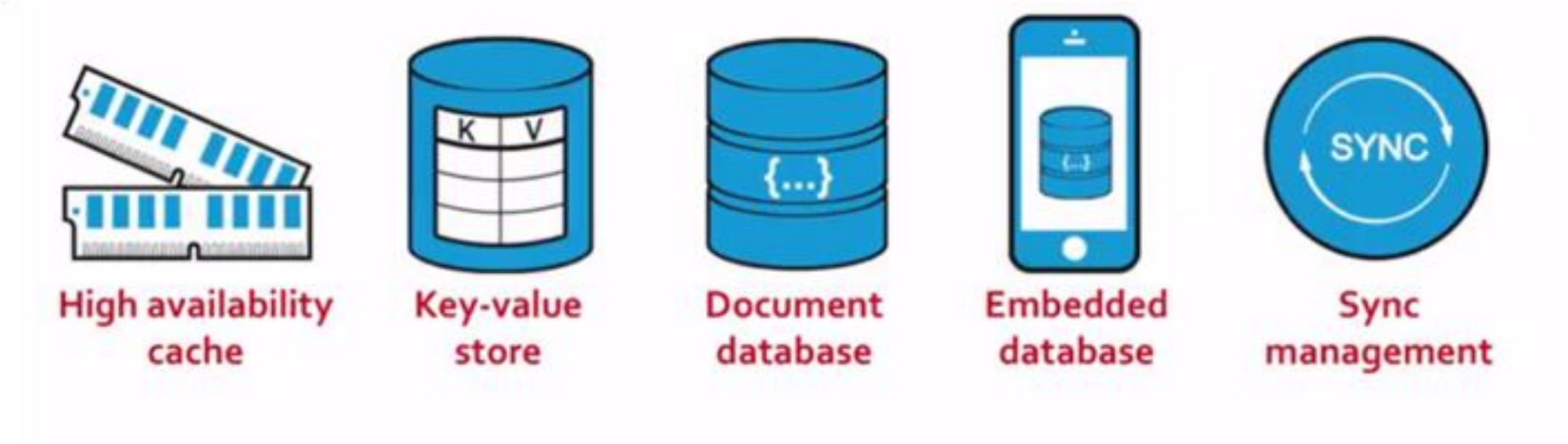
Couchbase

Consistent. Elastic. Scalable. Always available. Data storage.

Kelum Senanayake

What does Couchbase provide?

Multi-purpose operational capabilities support a broad range of use cases



History of Couchbase

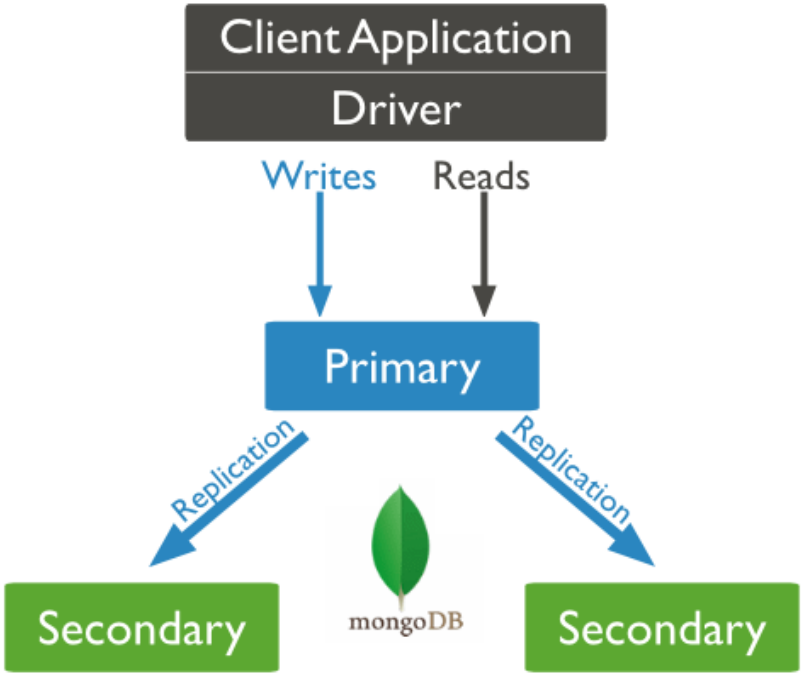
NorthScale developed a key-value storage engine

Apache CouchDB database project

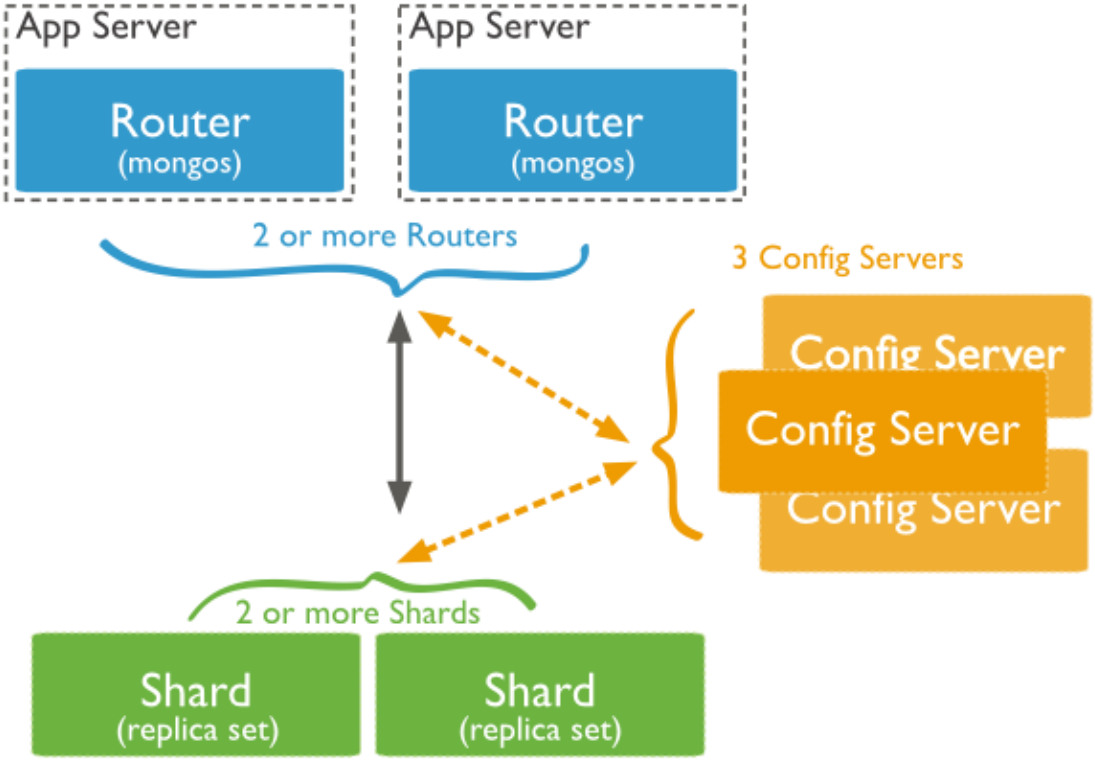
membase + CouchDB relax = Couchbase

Membase and CouchOne joined forces in February 2011 to create Couchbase, the first and only provider of comprehensive, end-to-end family of NoSQL database products

Any problem with MongoDB?

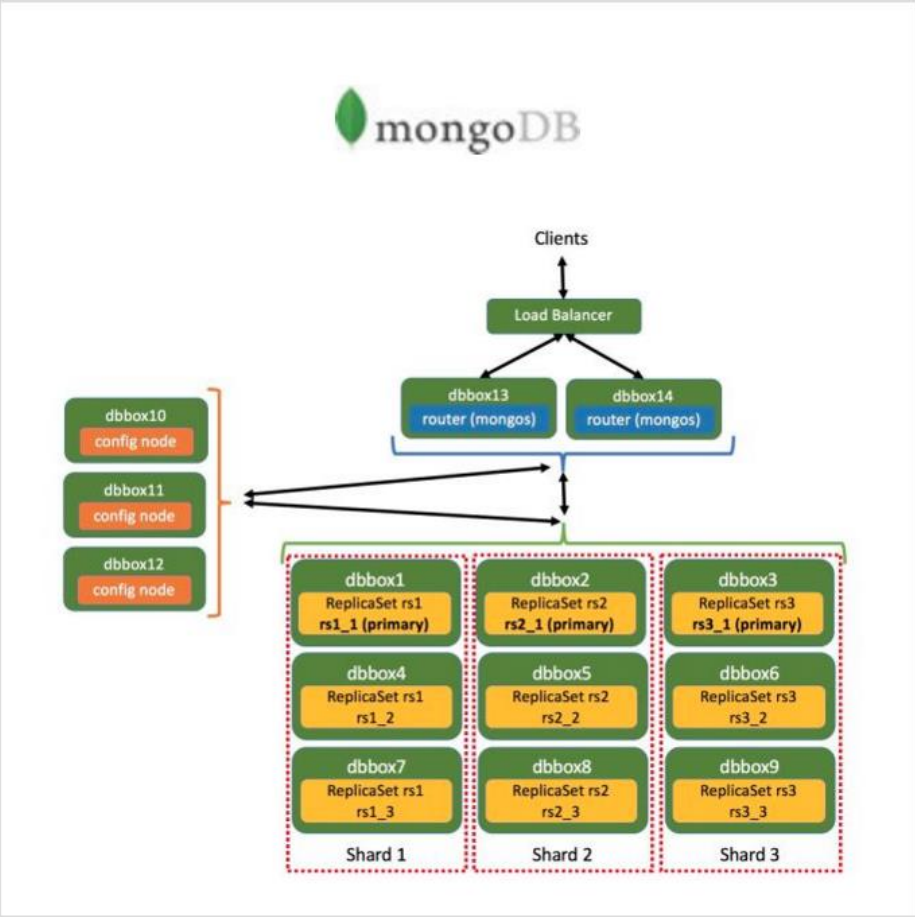


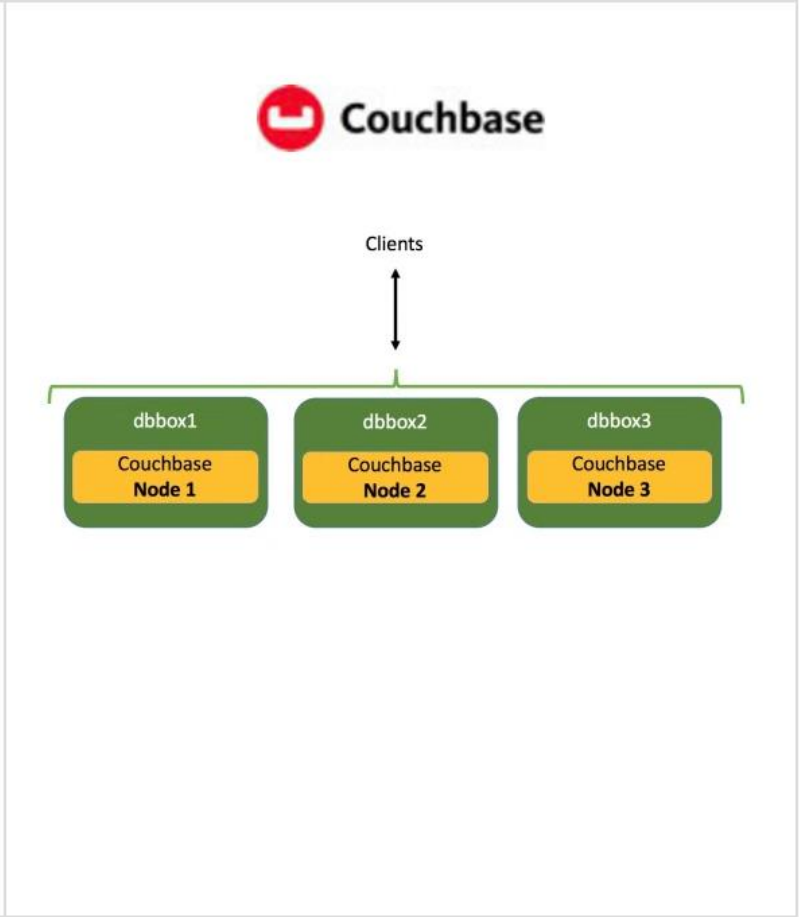
Any problem with MongoDB?



Server Utilization: Couchbase vs MongoDB

Server Utilization: Couchbase vs MongoDB

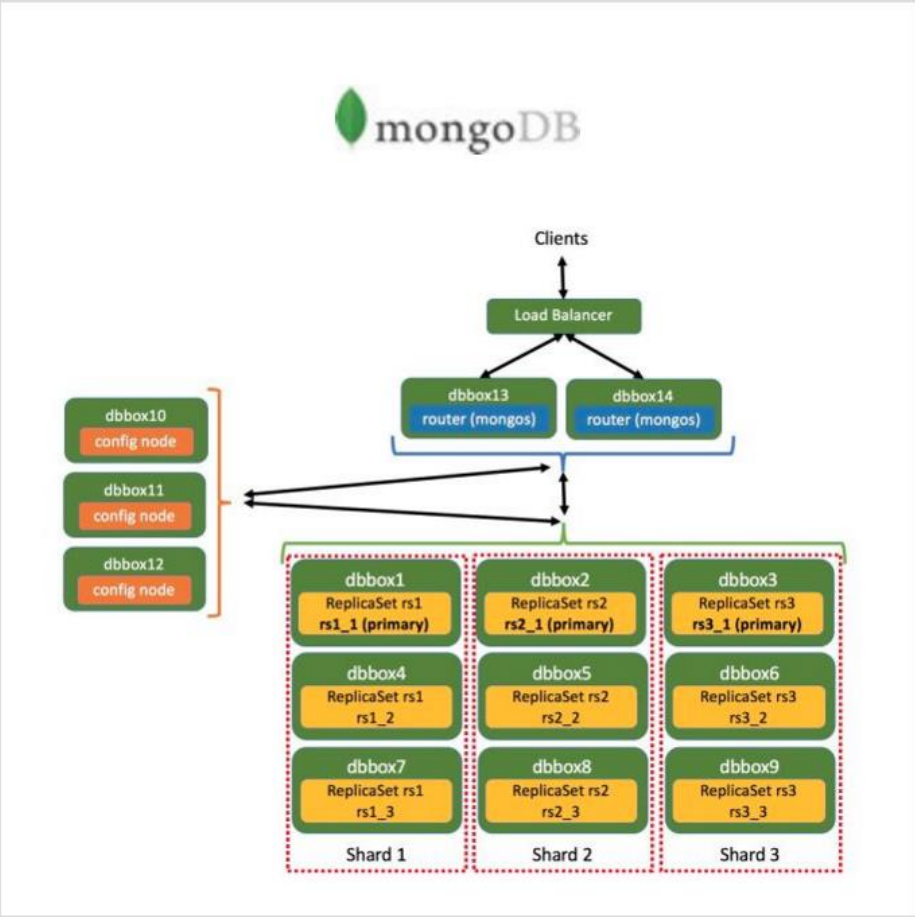


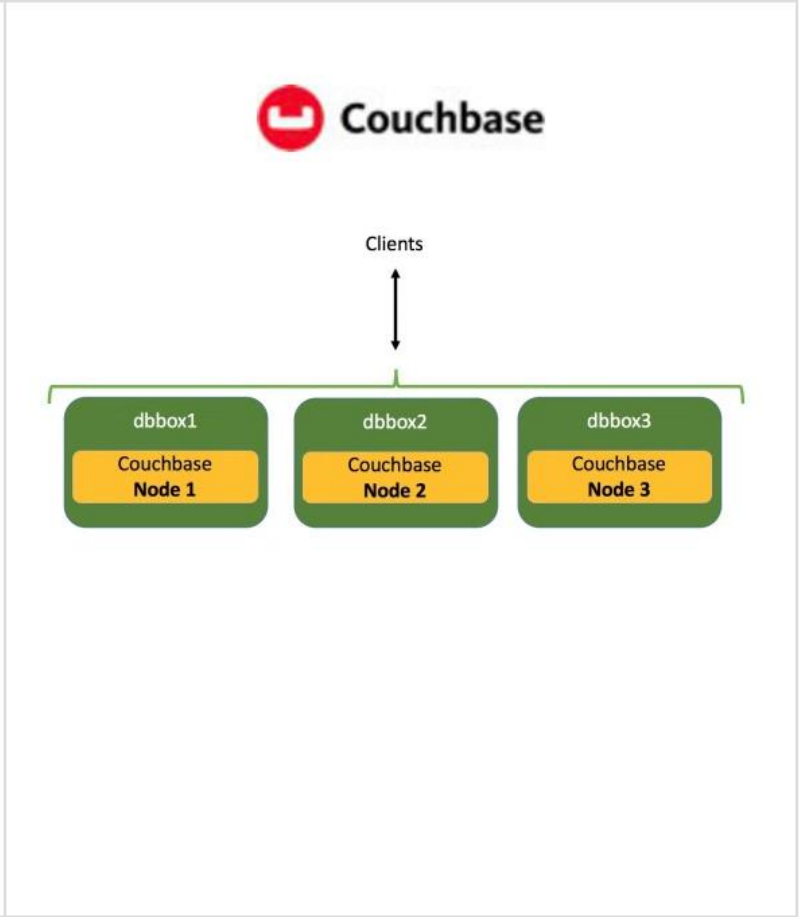


Server Utilization: Couchbase vs MongoDB

	mongoDB	Couchbase
Number of machines required	14	3
Load Balancer	1	Not required
Number of commands executed	128	27
Number of files edited	28	0

Server Utilization: Couchbase vs MongoDB





How does Couchbase store data?

Key-Value Pairs

```
2014-06-23-10:15am : 75F

2014-06-23-11:30am : 77F

2014-06-23-02:00pm : 82F
```

Documents

```
0001 :
{
    first_name : 'Rhonda'
    last_name : 'Red'
    language : 'EN'
    postal_code : 97203
}
```

Key ("Document ID")

Any string up to 250 bytes

Value

Any value up to 20MB

What does “document” mean?

Each key-identified value is a “document” regardless of size

Document IDs (“keys”) are manually created

- ✓ May be arbitrary or informative, but unique within a bucket.
- ✓ Hashed to determine the storage location

Value can be any type

- ✓ JSON encoded data, serialized object, XML, text, etc.

Each document includes metadata

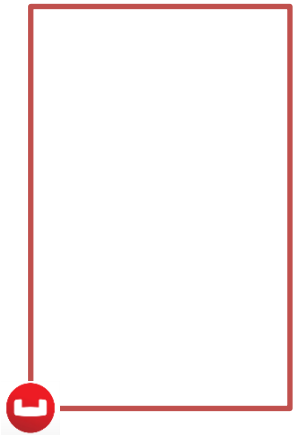
- ✓ Unique ID for optimistic concurrency (CAS)
- ✓ Optional expiration timestamp (TTL)
- ✓ Optional SDK specific flags (ex: type, format)

What are main architectural structures?

What are main architectural structures?

Node

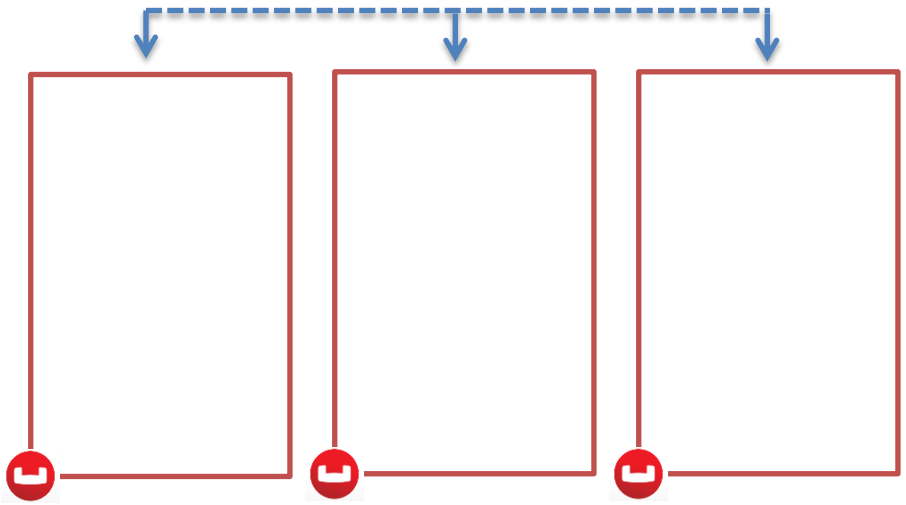
A Couchbase server instance



What are main architectural structures?

Node
A Couchbase server instance

Cluster
A scalable, networked set of nodes, sharing distributed buckets

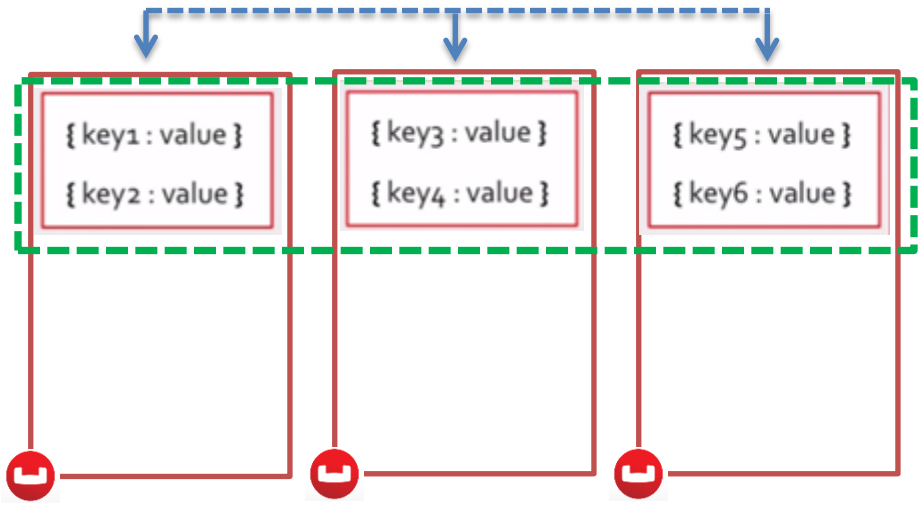


What are main architectural structures?

Node
A Couchbase server instance

Cluster
A scalable, networked set of nodes, sharing distributed buckets

Bucket
A logical key space of uniquely keyed documents, evenly distributed across a cluster



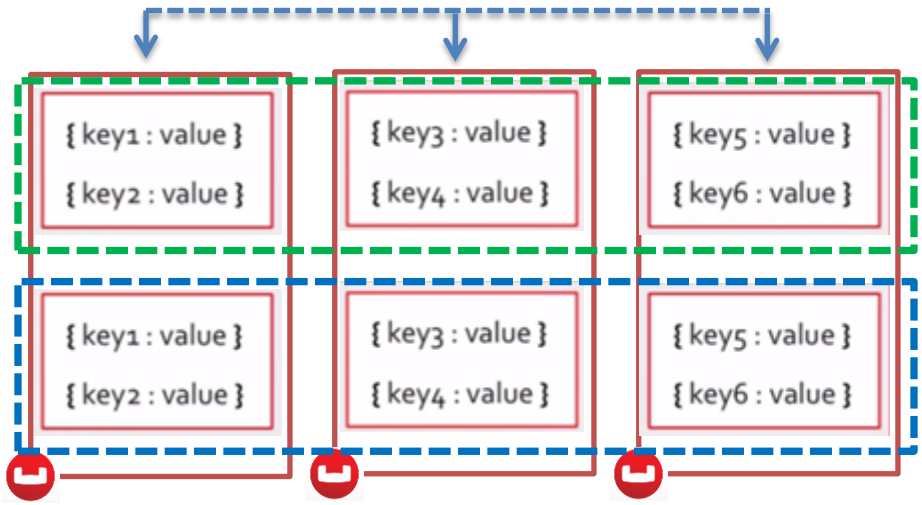
What are main architectural structures?

Node
A Couchbase server instance

Cluster
A scalable, networked set of nodes, sharing distributed buckets

Bucket
A logical key space of uniquely keyed documents, evenly distributed across a cluster

Document
A uniquely keyed value within a particular bucket



How do you access data?

Client applications have four ways to access data

Read/Write documents by their specific key

- ✓ Extremely fast due to working set cache management
- ✓ Reads and writes are immediately consistent

MapReduce Views

- ✓ Distributed secondary indexes, built via map-reduce
- ✓ Accessed by REST base Views API

N1QL ("Nickel") Queries

- ✓ SQL superset for indexing and querying JSON documents

Full text search

- ✓ Couchbase FTS (Developer preview on v4.5)

What is the high level architecture?

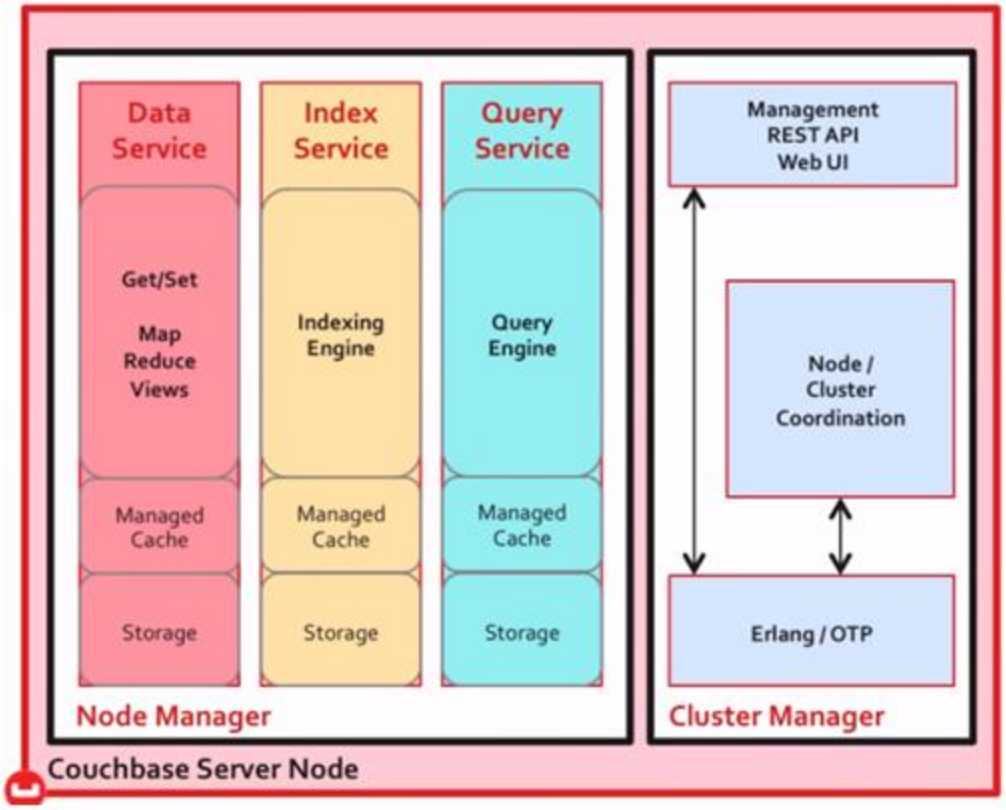
Couchbase Server nodes are identical

Two core components

- ✓ Cluster Manager
- ✓ Node Manager

Three independent services

- ✓ Data Service
- ✓ Index Service
- ✓ Query Service



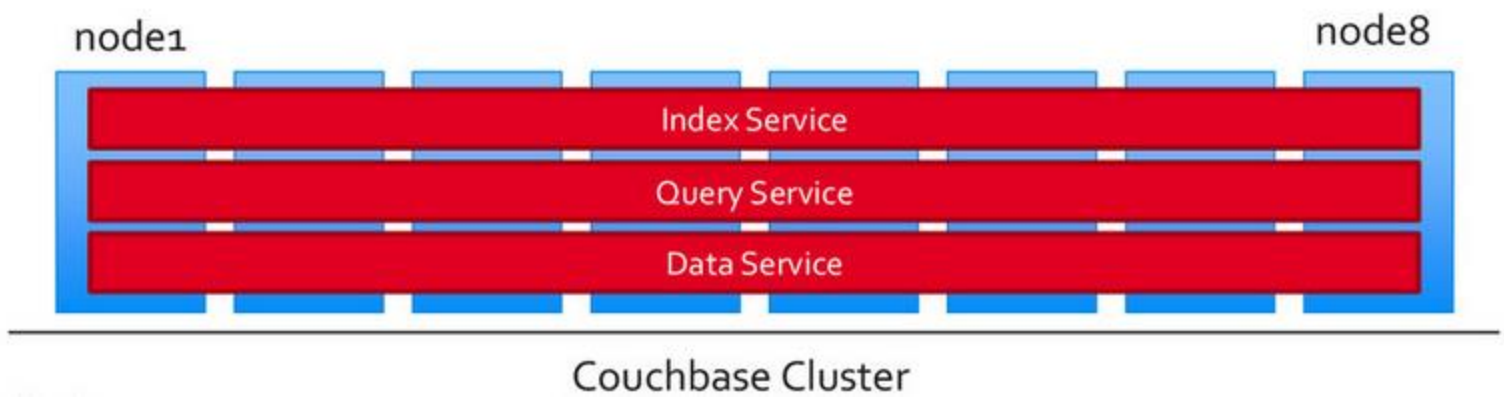
Scalability Model Today

Homogeneous scaling

- Each node gets a slice of the workload
- Simple to do…

But..

- Workload compete and interfere with each other
- Cannot fine tune each workload
  - **Query:** Query is a CPU heavy operation
  - **Index:** Index service is disk intensive
  - **Data:** Data nodes require more memory

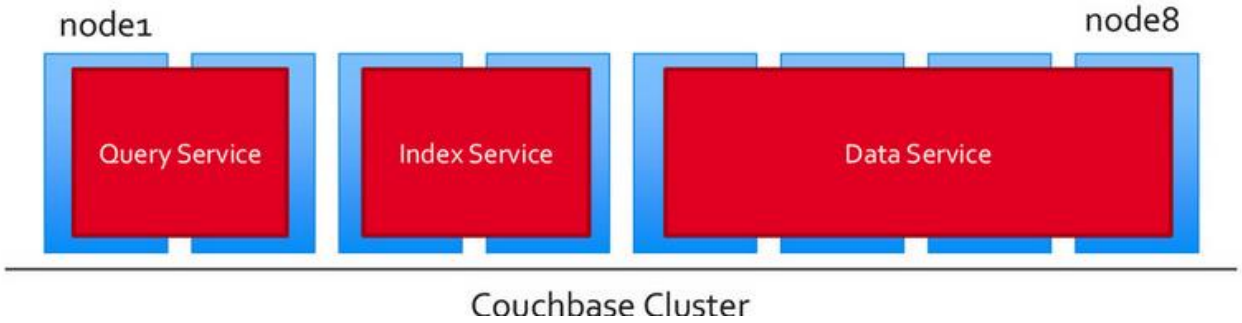


Modern Architecture (Multi-Dimensional Scaling)

Modern Architecture (Multi-Dimensional Scaling)

What is Multi-Dimensional Scaling?

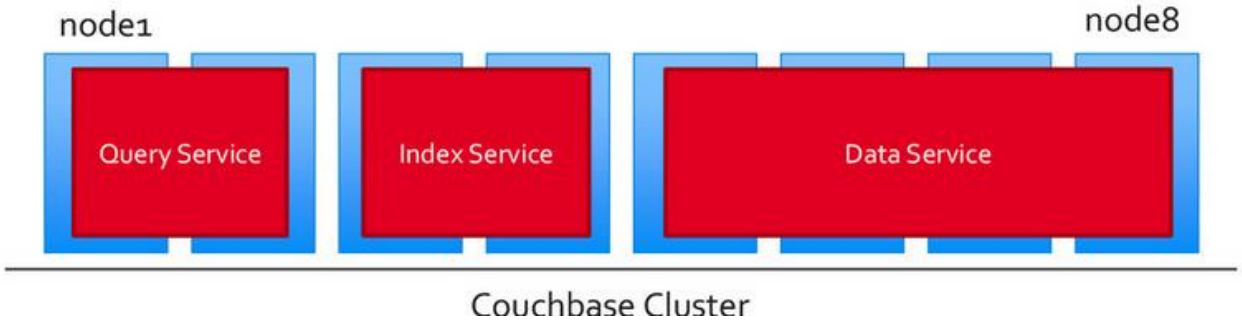
MDS is the architecture that enables independent scaling of data, query and indexing workloads

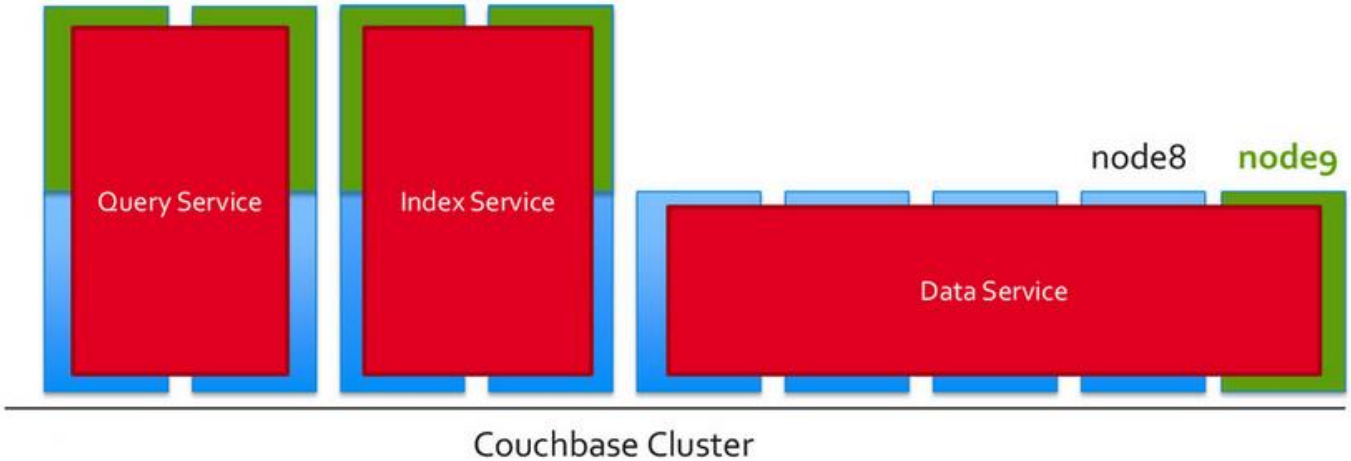


Modern Architecture (Multi-Dimensional Scaling)

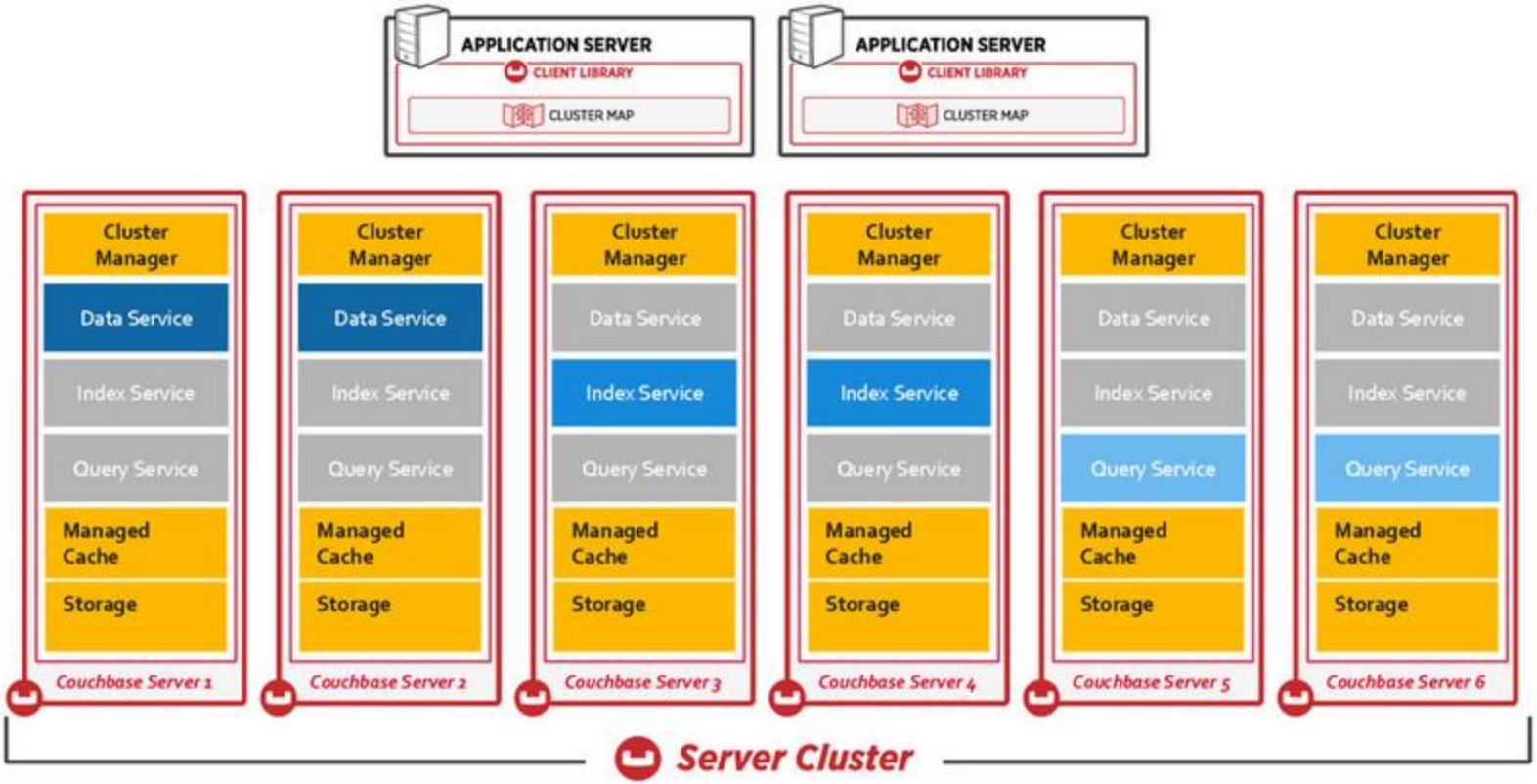
What is Multi-Dimensional Scaling?

MDS is the architecture that enables independent scaling of data, query and indexing workloads

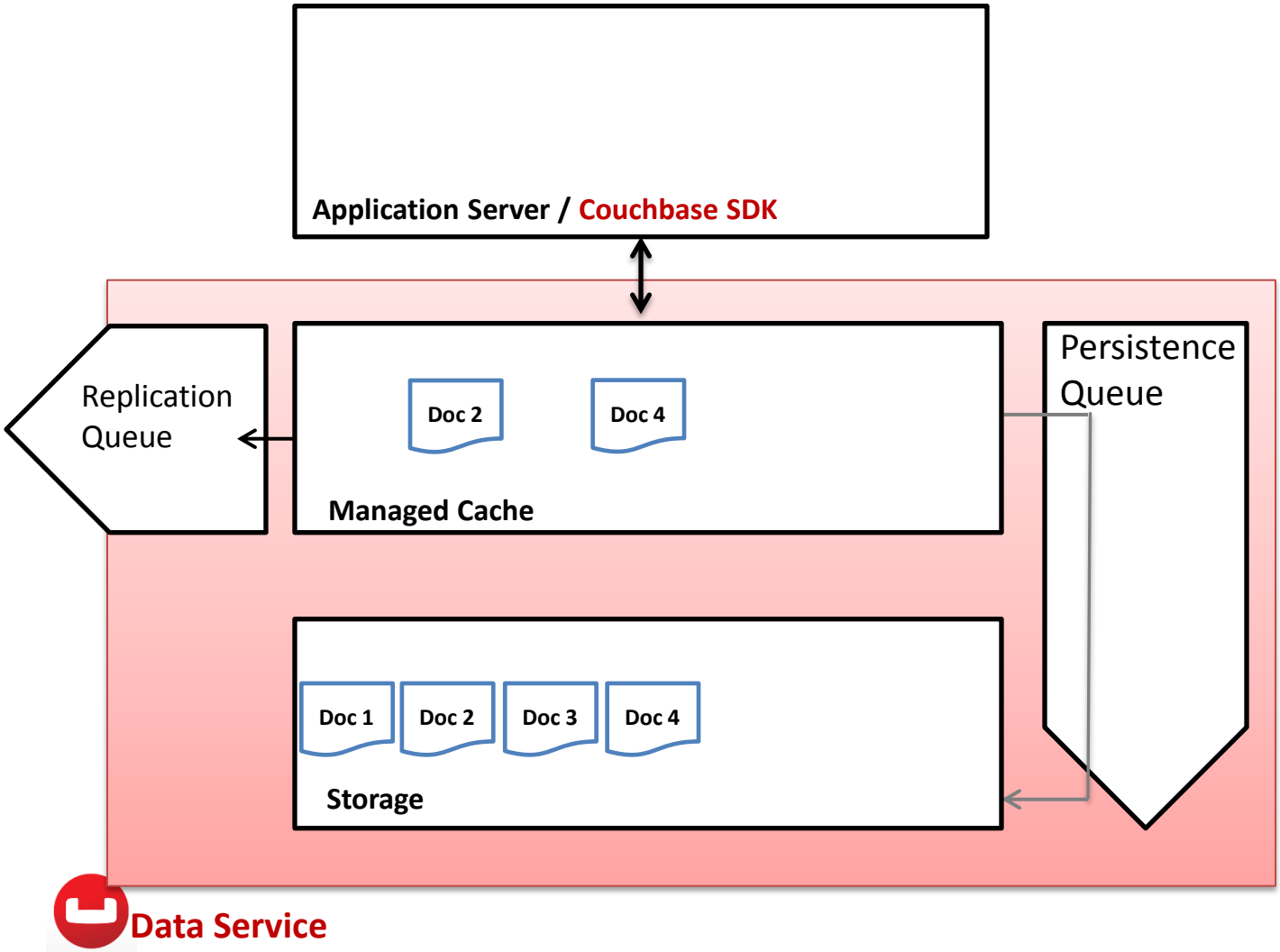




Full Cluster Architecture

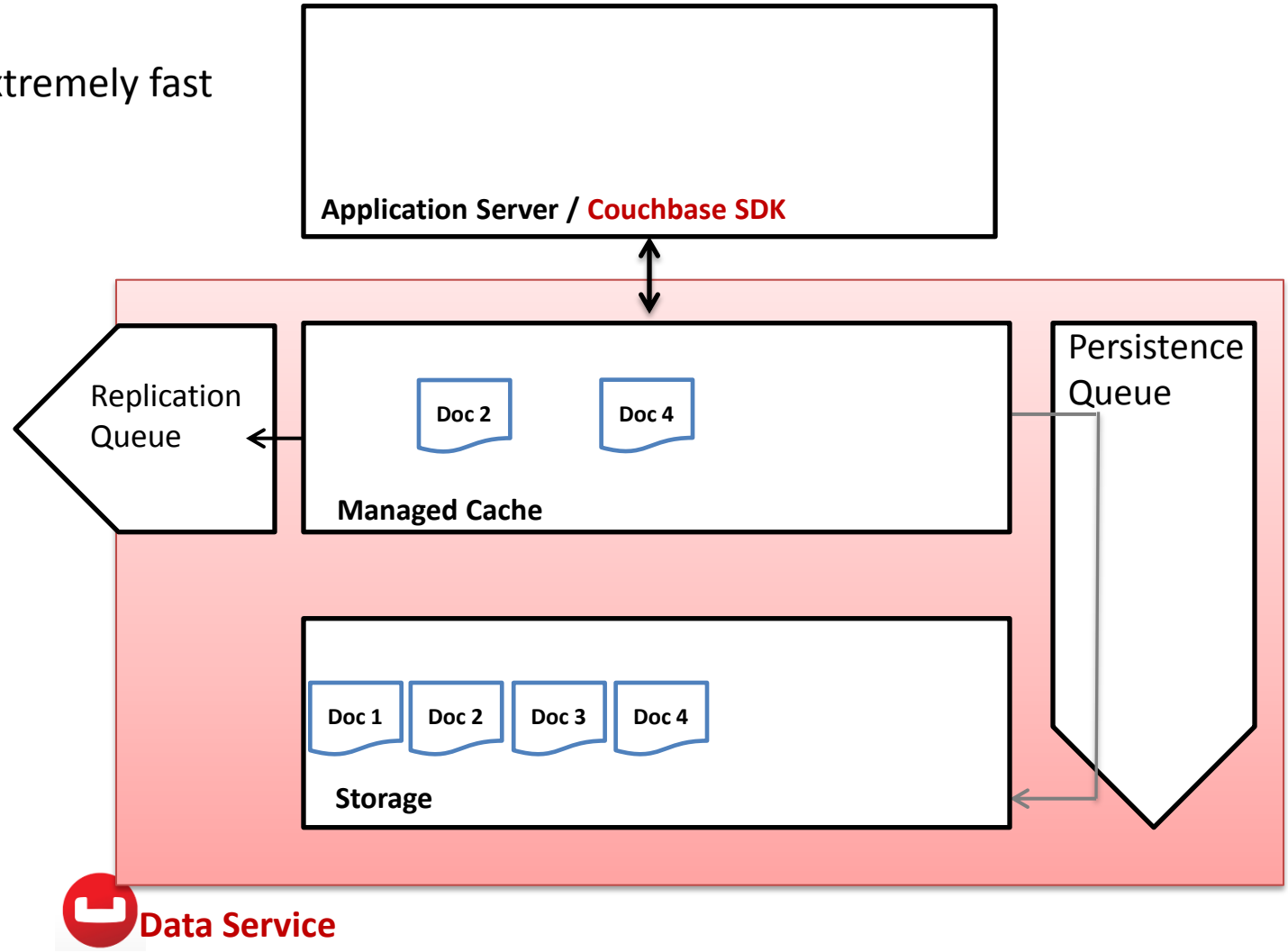


How does a **get** operate?



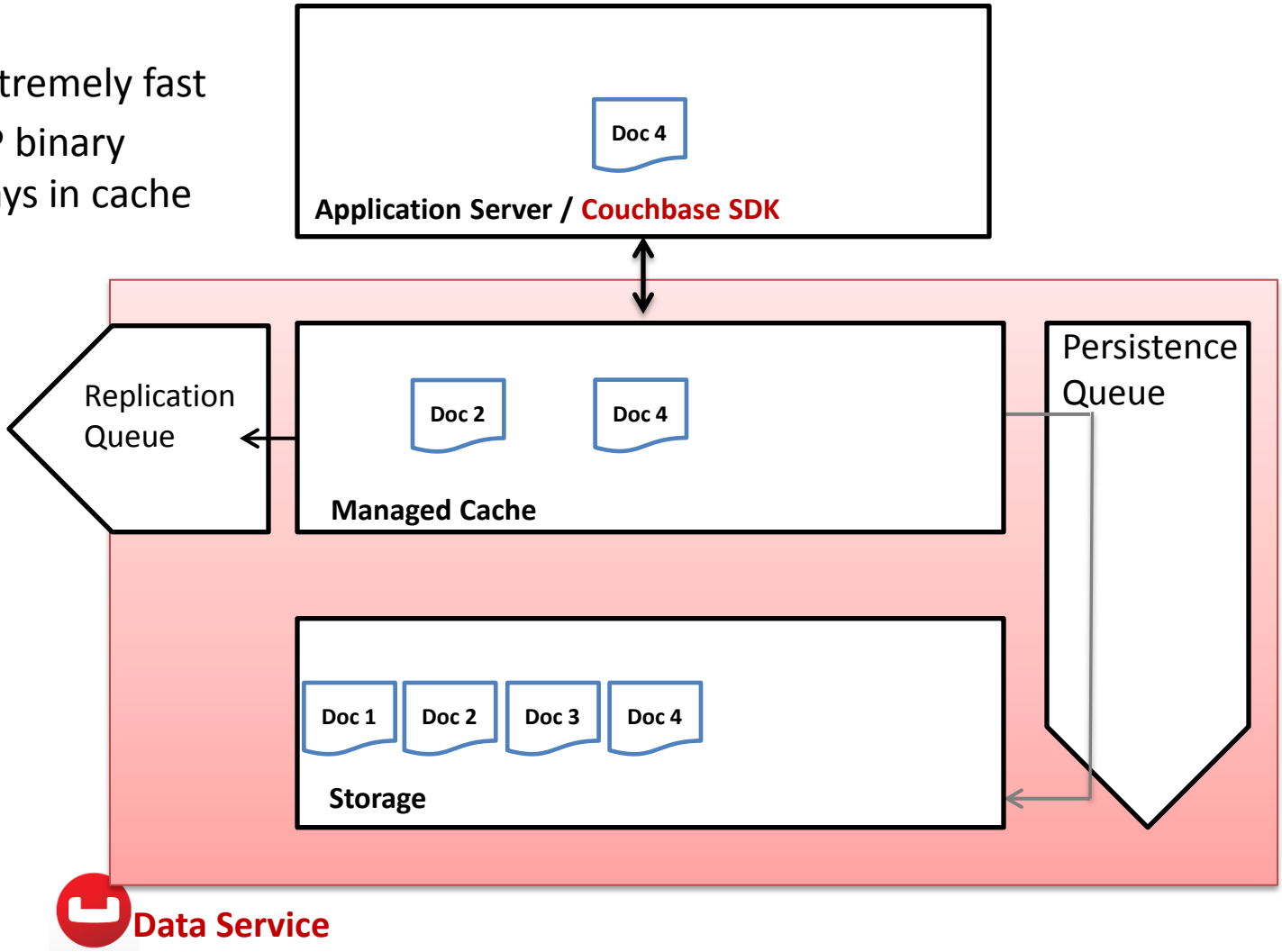
How does a **get** operate?

- ✓ Cache gets are extremely fast



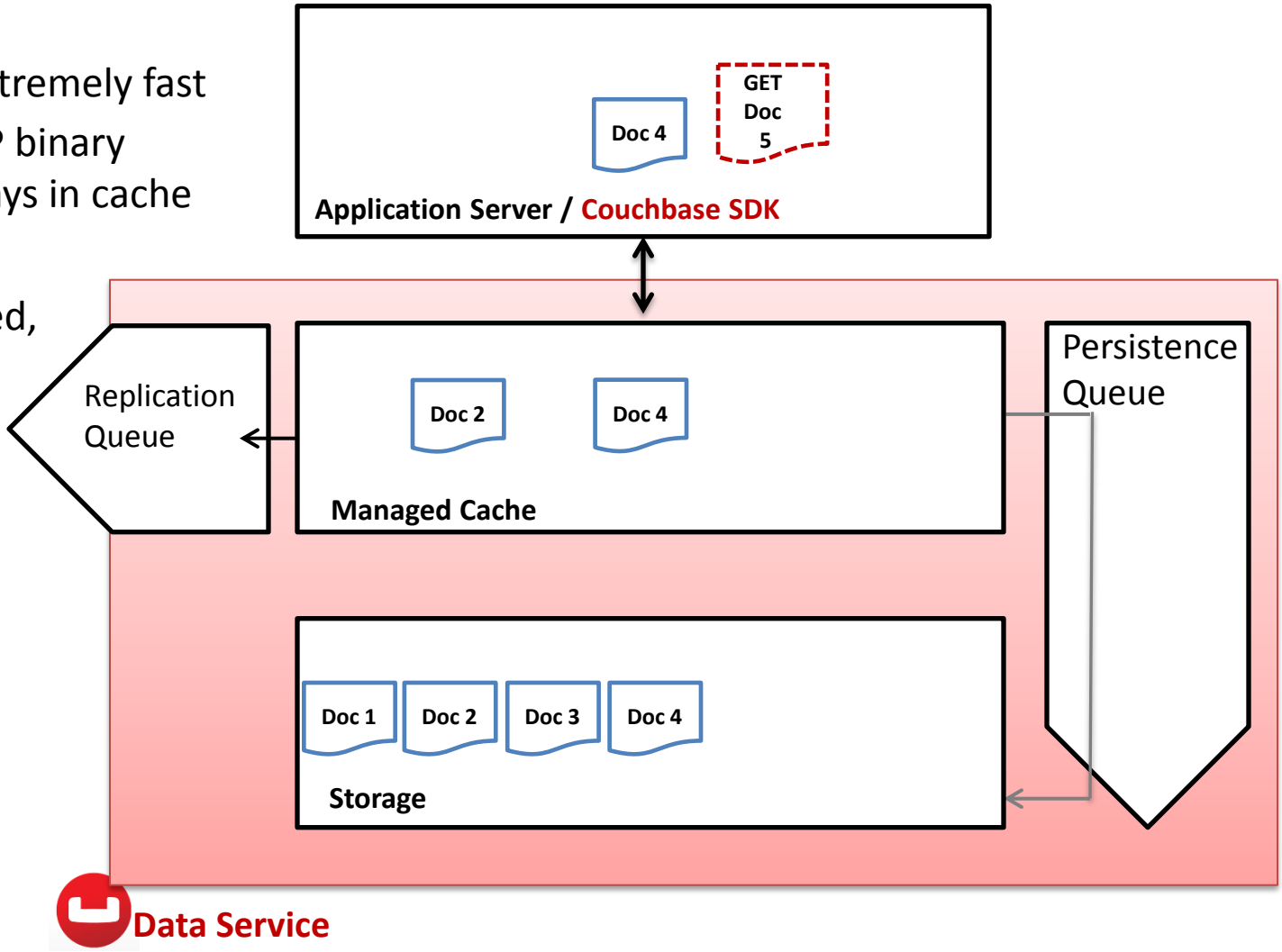
How does a **get** operate?

- ✓ Cache gets are extremely fast
- ✓ Connection is TCP binary
- ✓ Common docs stays in cache



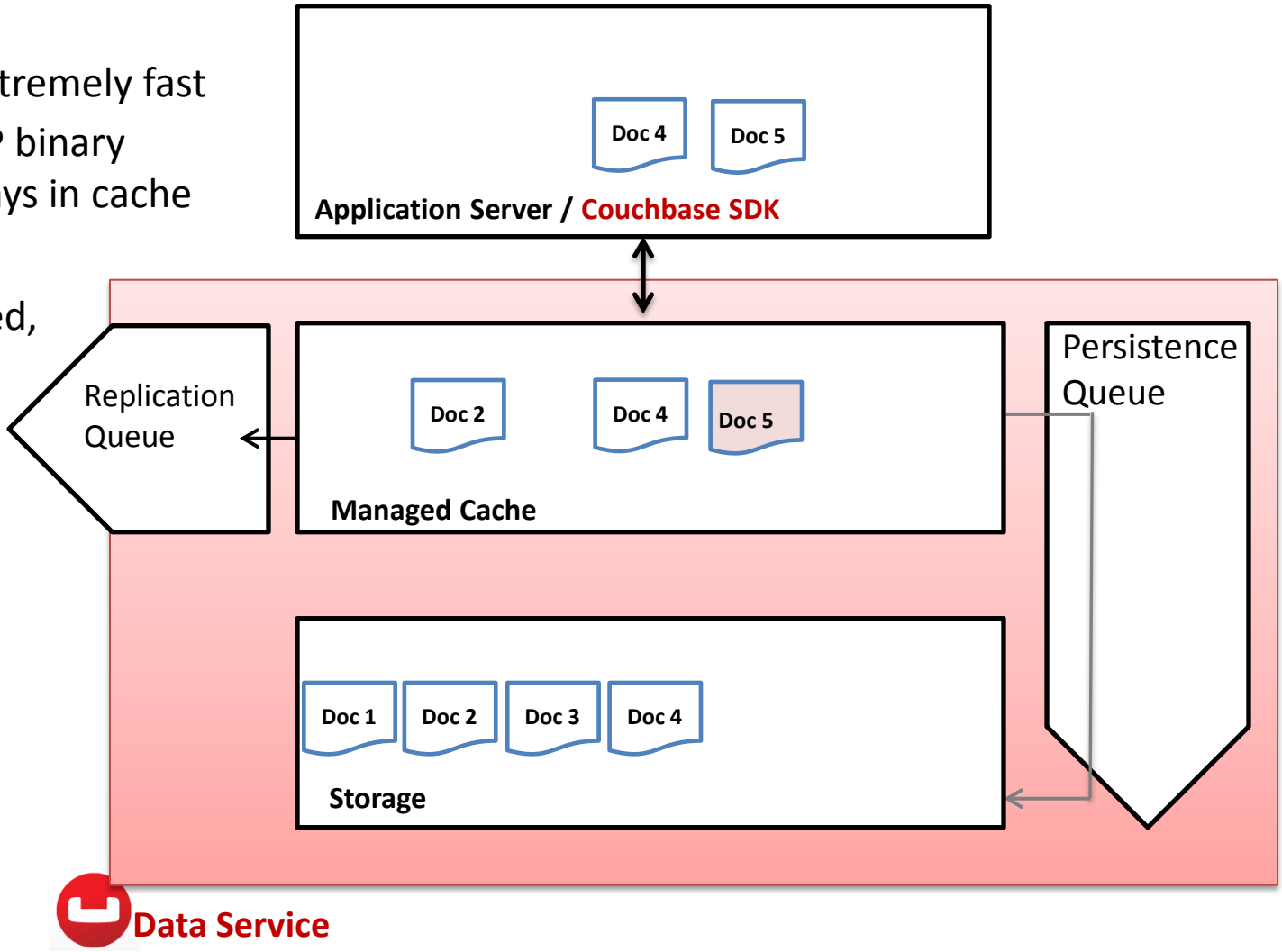
How does a **get** operate?

- ✓ Cache gets are extremely fast
- ✓ Connection is TCP binary
- ✓ Common docs stays in cache
- ✓ Un-cached docs retrieved, returned, then cached



How does a **get** operate?

- ✓ Cache gets are extremely fast
- ✓ Connection is TCP binary
- ✓ Common docs stays in cache
- ✓ Un-cached docs retrieved, returned, then cached



How is the cache managed?

NRU (Not Recently Used) score is maintained by each cache item

Nodes configurable for **value-only** or **full ejection**

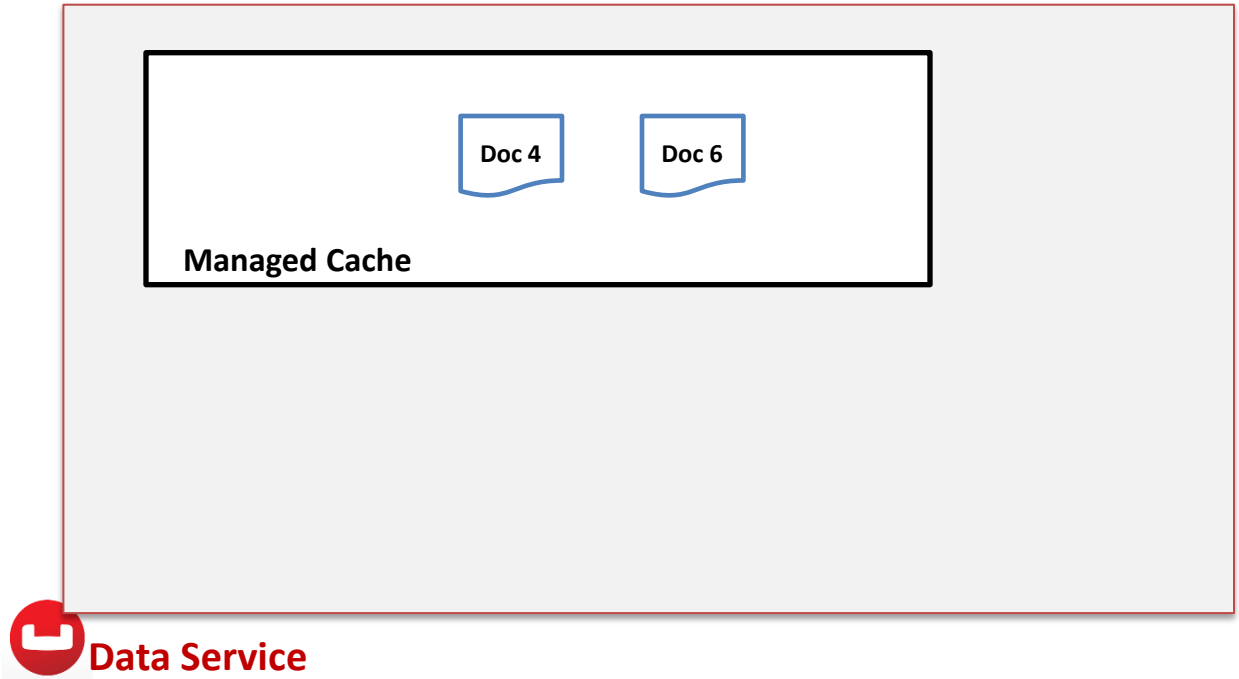
Value-only

- ✓ Max lookup speed
- ✓ Max memory use and slow warm-up time

Full-ejection

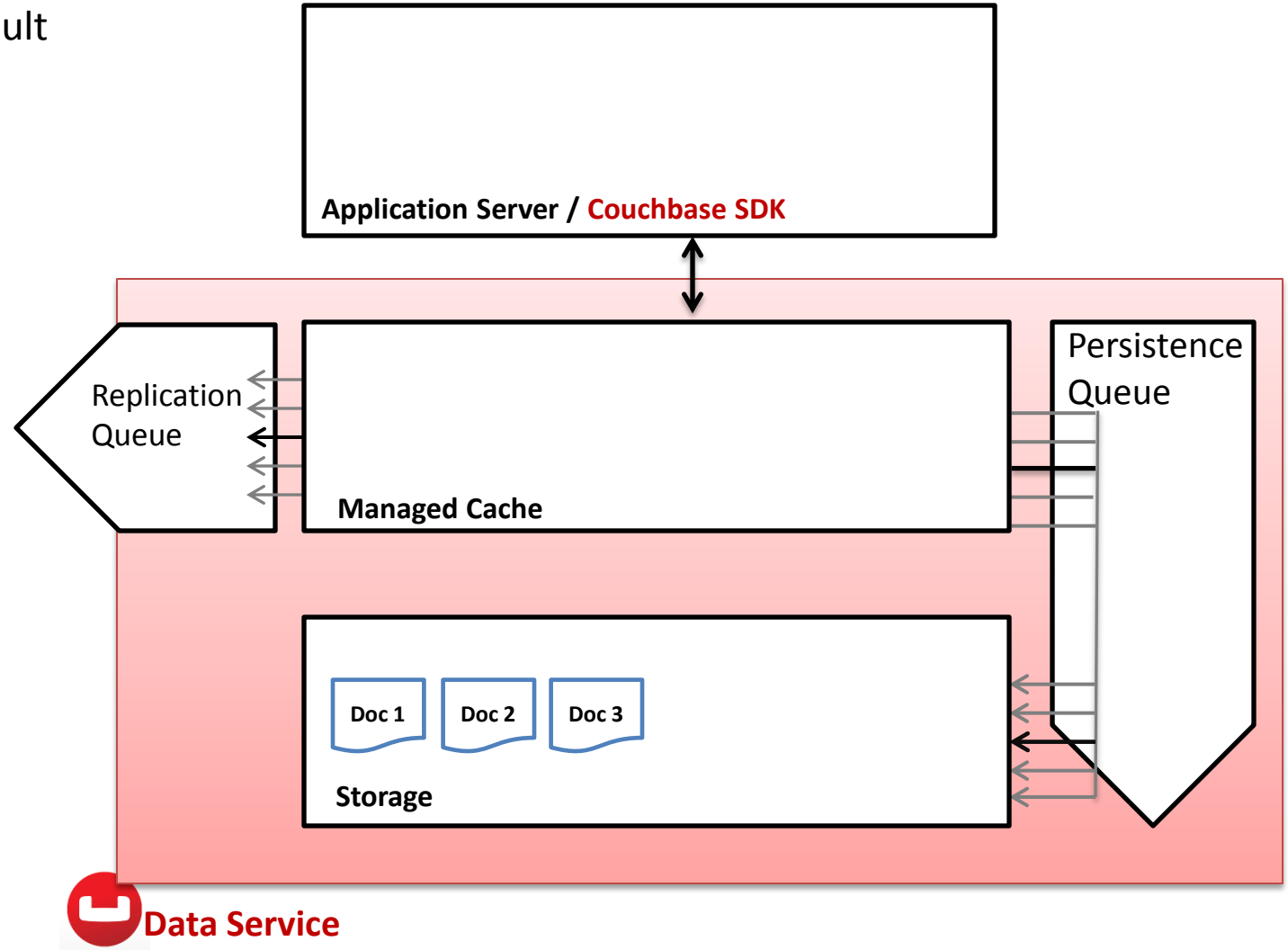
- ✓ Slower lookup speed
- ✓ Lower memory use

Best choice varies by use case



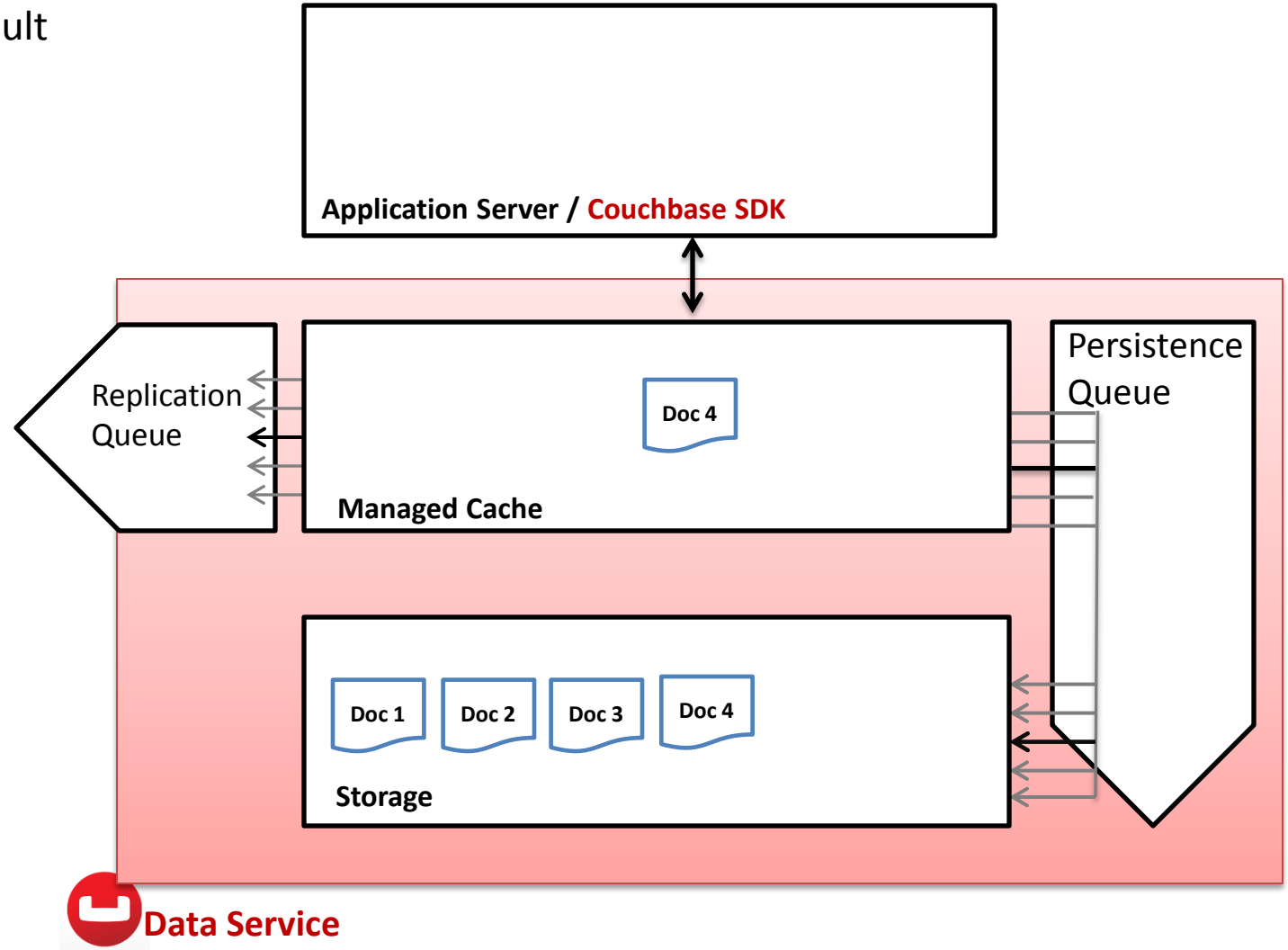
How does a **set** operate?

Sets are async by default



How does a **set** operate?

Sets are async by default

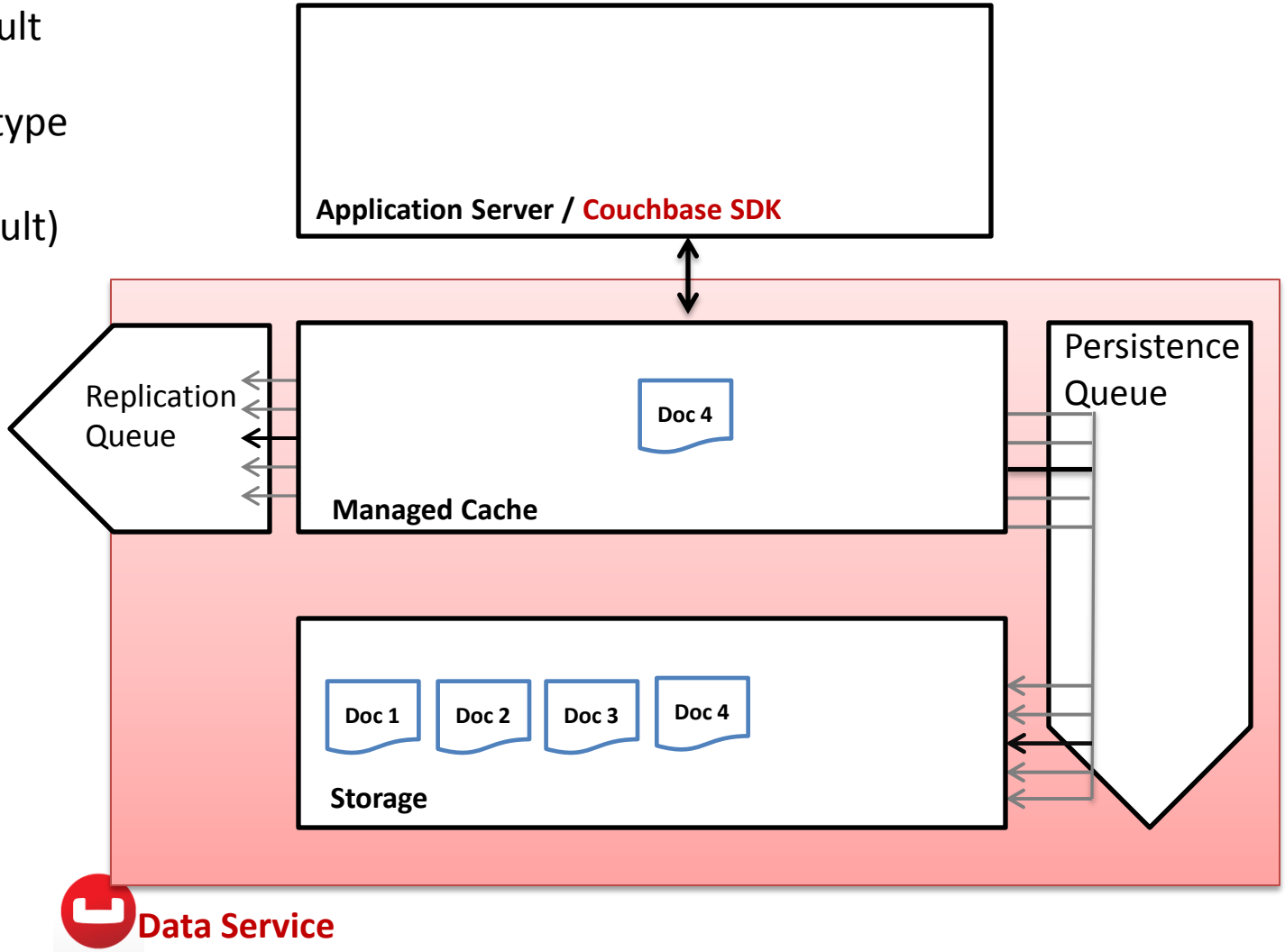


How does a **set** operate?

Sets are async by default

Choose acknowledge type *per write*

- ✓ when in RAM (default)
- ✓ when in storage
- ✓ when replicated



How does a **set** operate?

Sets are async by default

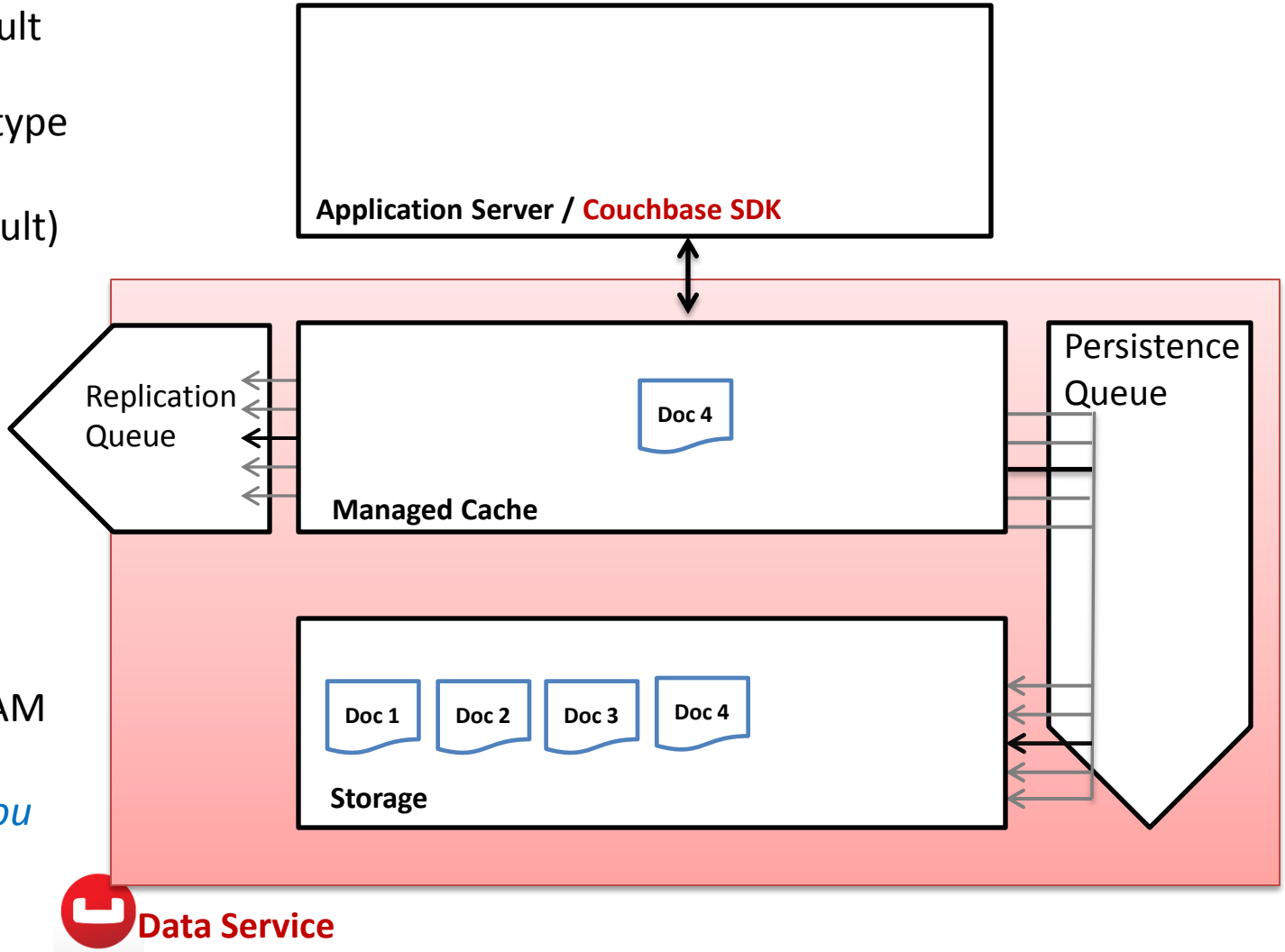
Choose acknowledge type *per write*

- ✓ when in RAM (default)
- ✓ when in storage
- ✓ when replicated

Replication

- ✓ 1, 2 or 3 nodes
- ✓ very fast, RAM to RAM

Nothing waits unless you choose



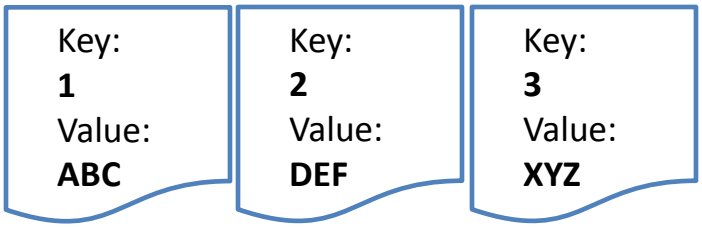
Why Couchbase server so fast?

Gets and sets are served primarily from cache

Why Couchbase server so fast?

Gets and sets are served primarily from cache

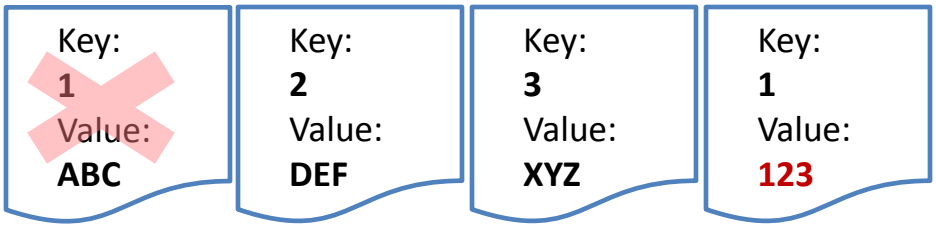
Disk writes are append only



Why Couchbase server so fast?

Gets and sets are served primarily from cache

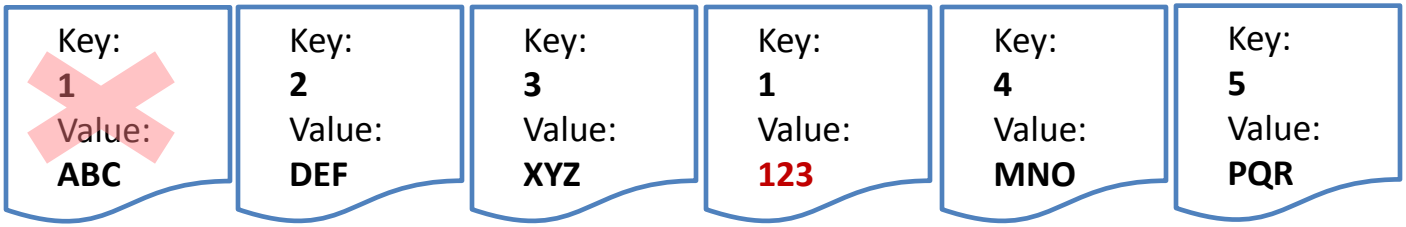
Disk writes are append only



Why Couchbase server so fast?

Gets and sets are served primarily from cache

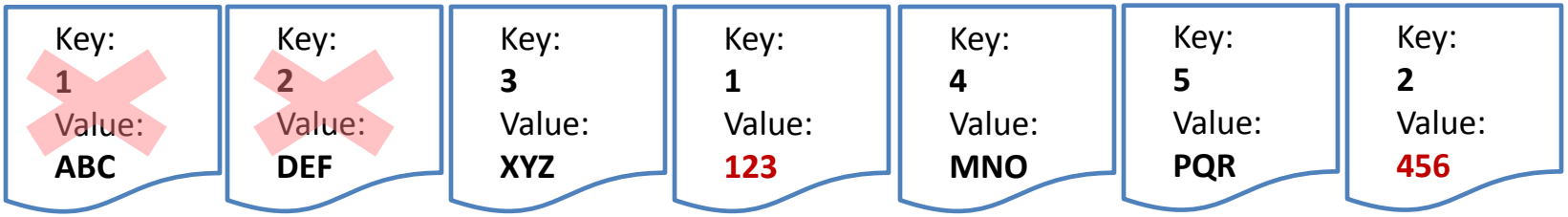
Disk writes are append only



Why Couchbase server so fast?

Gets and sets are served primarily from cache

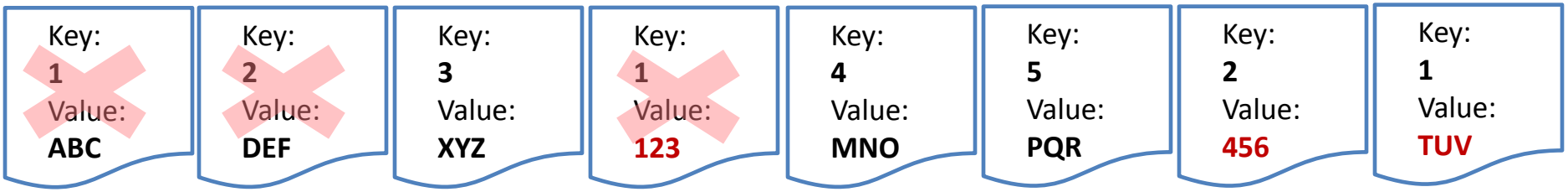
Disk writes are append only



Why Couchbase server so fast?

Gets and sets are served primarily from cache

Disk writes are append only

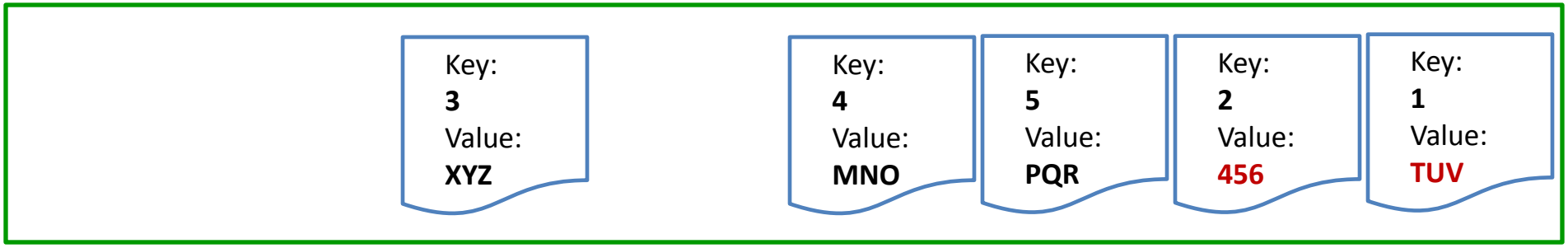


Why Couchbase server so fast?

Gets and sets are served primarily from cache

Disk writes are append only

Data files are periodically compacted



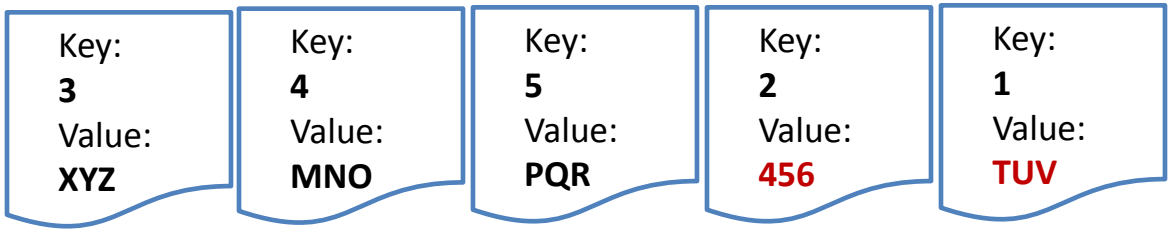
Why Couchbase server so fast?

Gets and sets are served primarily from cache

Disk writes are append only

Data files are periodically compacted

- ✓ Compacted replacement created and put online
- ✓ Zero impact on read/write ops due to memory-focused architecture



How do MapReduce Views work?

View are secondary indexes defined by map functions deployed in the Data Service

```
function(doc, meta) {
  if(doc.sales > 100000) {
    emit(doc.city, [doc.name, doc.sales]);
  }
}
```

- ✓ Written in JavaScript
- ✓ Processed by V8 JS Engine
- ✓ Get results via REST API port 8092
- ✓ Numerous query parameters supported for filtering results

```
https://[localhost]:8092/[bucket-name]/_design/
[ddoc-name]/_view/[view-name]?limit=10
```

How does N1QL works?

SQL for multi-dimensional,
flexible data..

- ✓ SELECT, INSERT, UPDATE, DELETE
- ✓ JOIN, WHERE, HAVING, GROUP BY
- ✓ CREATE INDEX, DROP INDEX
- ✓ MIN, MAX, COUNT
- ✓ UNION, INTERSECT, EXCEPT
- ✓ NEST, UNNEST

…more

- Client code simplification
- Ad hoc queries
- Prepared statements

http://query.couchbase.com/tutorial

```
SELECT count(*), state
FROM customer
WHERE customer.ccInfo.cardType = "discover"
GROUP BY customer.state
ORDER BY customer.state
LIMIT 5 OFFSET 5
```

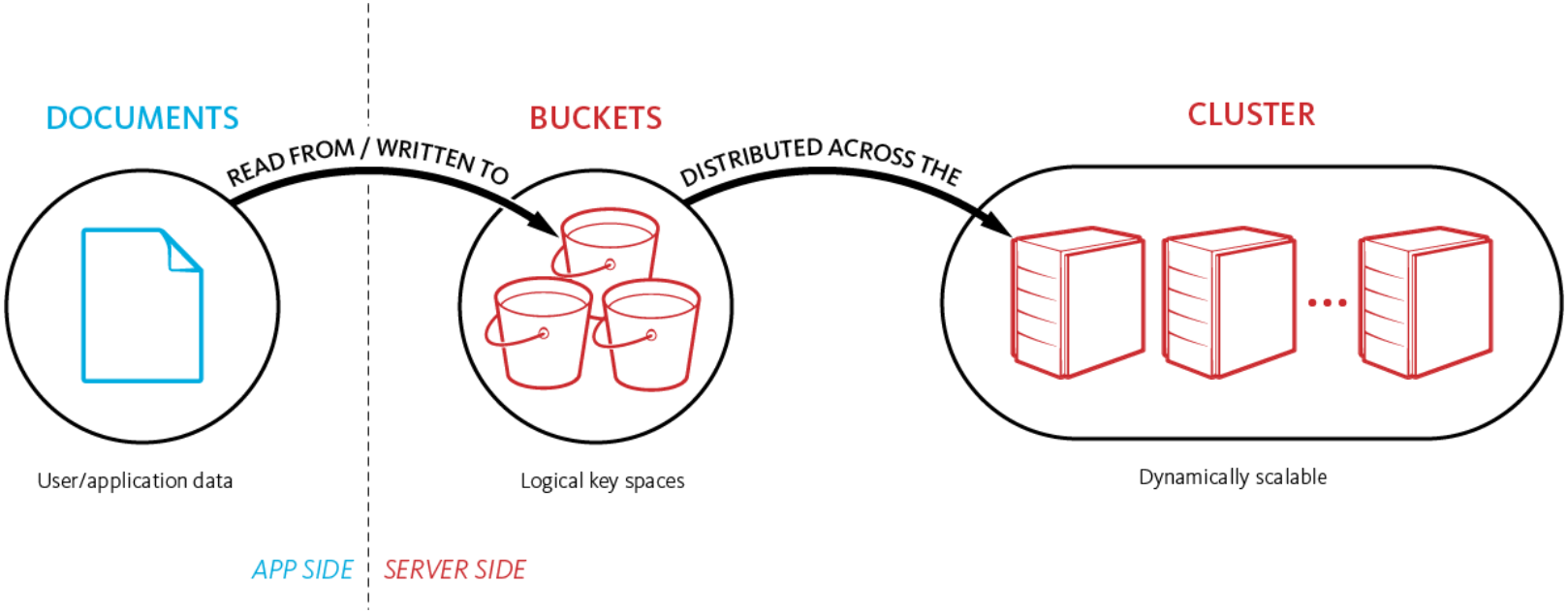
```
SELECT  count(DISTINCT customerId)
FROM purchases
```

```
SELECT
AVG(reviews.rating) / 5 as normalizedRating,
ROUND((avg(reviews.rating) / 5), 2) as
roundedRating,
TRUNC((avg(reviews.rating) / 5), 3) as
truncRating
FROM reviews AS reviews
WHERE reviews.customerId = "customer62"
```

What is a Data Bucket?

A logical container of uniquely keyed documents

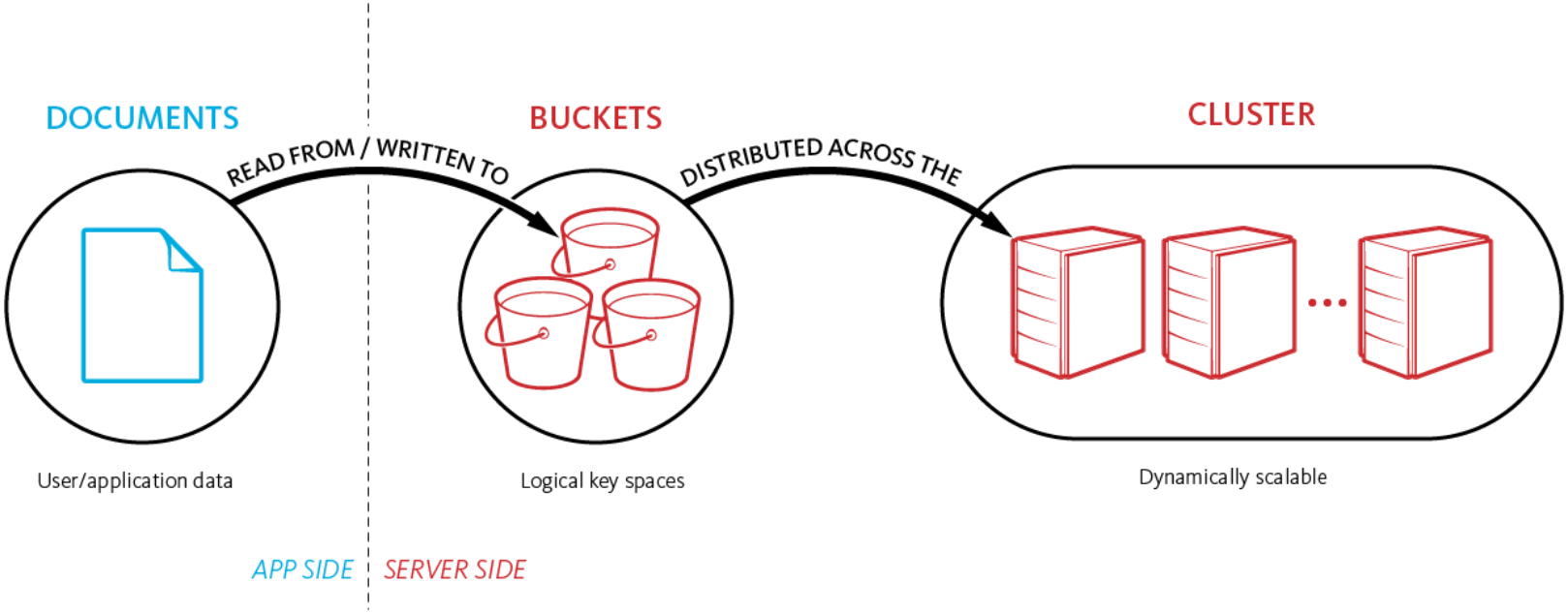
- ✓ Keyspace
- ✓ Database



What is a Data Bucket?

A logical container of uniquely keyed documents

- ✓ Keyspace
- ✓ Database



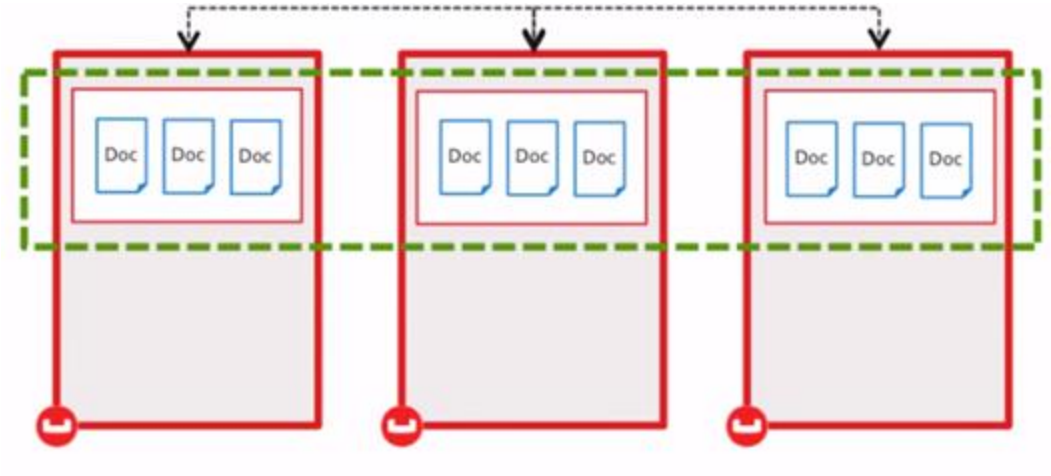
So, what equates to a “table”?

What is a Virtual Bucket?

One Organizational segment of a Data Bucket

Each bucket is divided into 1024 segments, evenly distributed across all nodes in the cluster

- ✓ *virtual buckets ("vBuckets")*



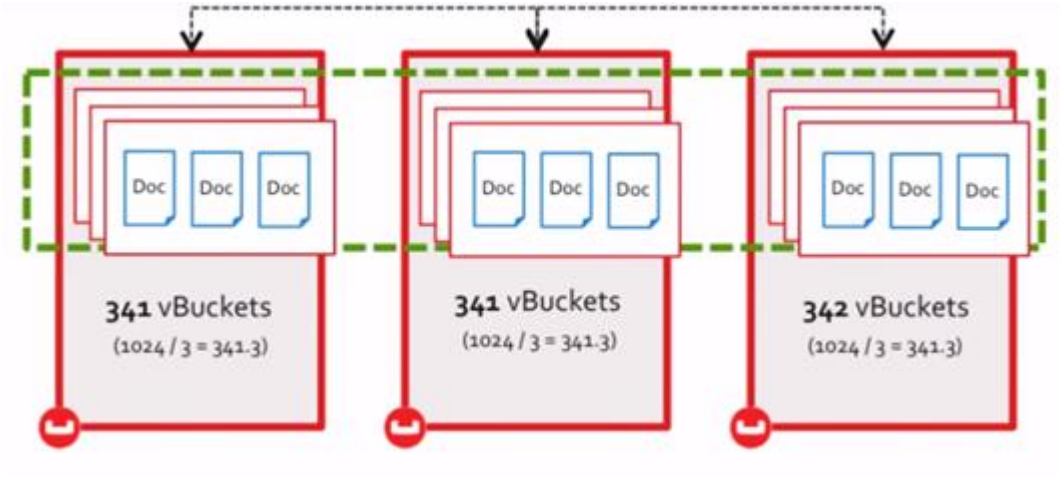
What is a Virtual Bucket?

One Organizational segment of a Data Bucket

Each bucket is divided into 1024 segments, evenly distributed across all nodes in the cluster

- ✓ *virtual buckets ("vBuckets")*

As nodes join/leave cluster, vBuckets adjust automatically



What is a Virtual Bucket?

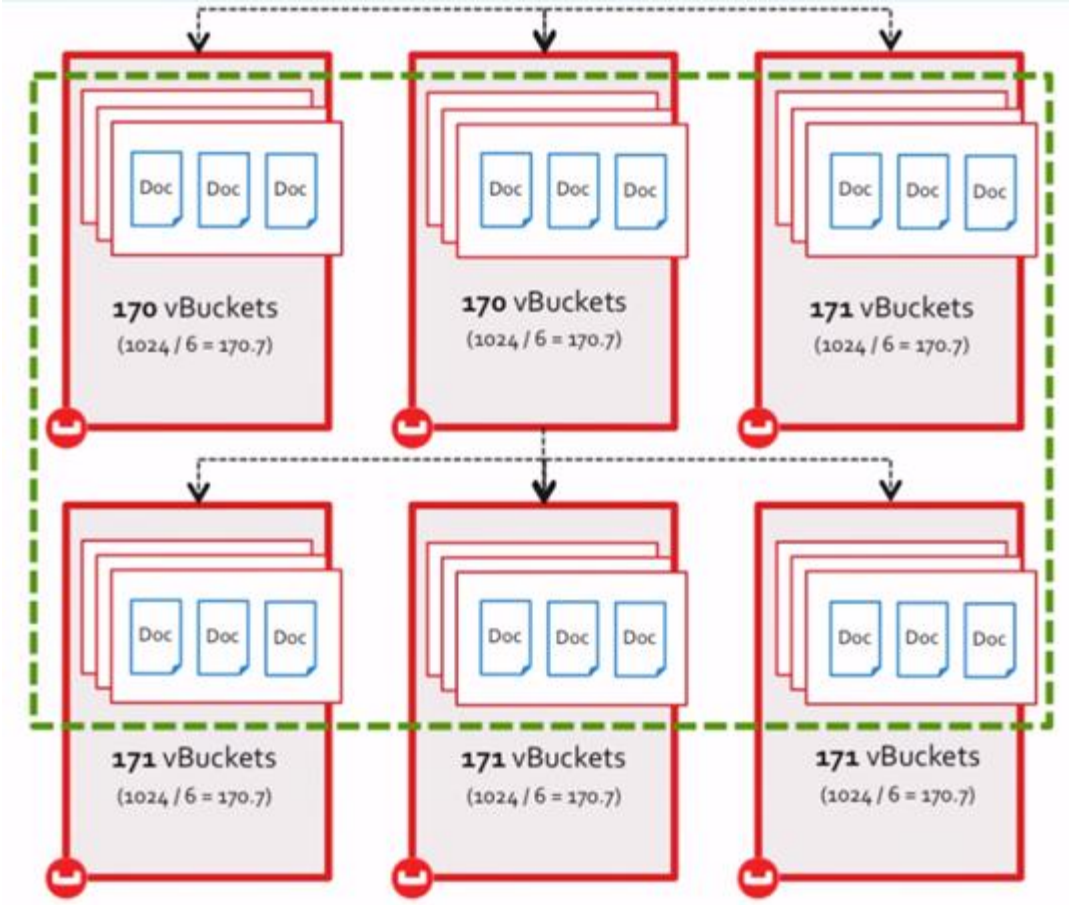
One Organizational segment of a Data Bucket

Each bucket is divided into 1024 segments, evenly distributed across all nodes in the cluster

- ✓ *virtual buckets ("vBuckets")*

As nodes join/leave cluster, vBuckets adjust automatically

Location and number of vBuckets is tracked by the Couchbase SDK *cluster map*



What is the cluster map?

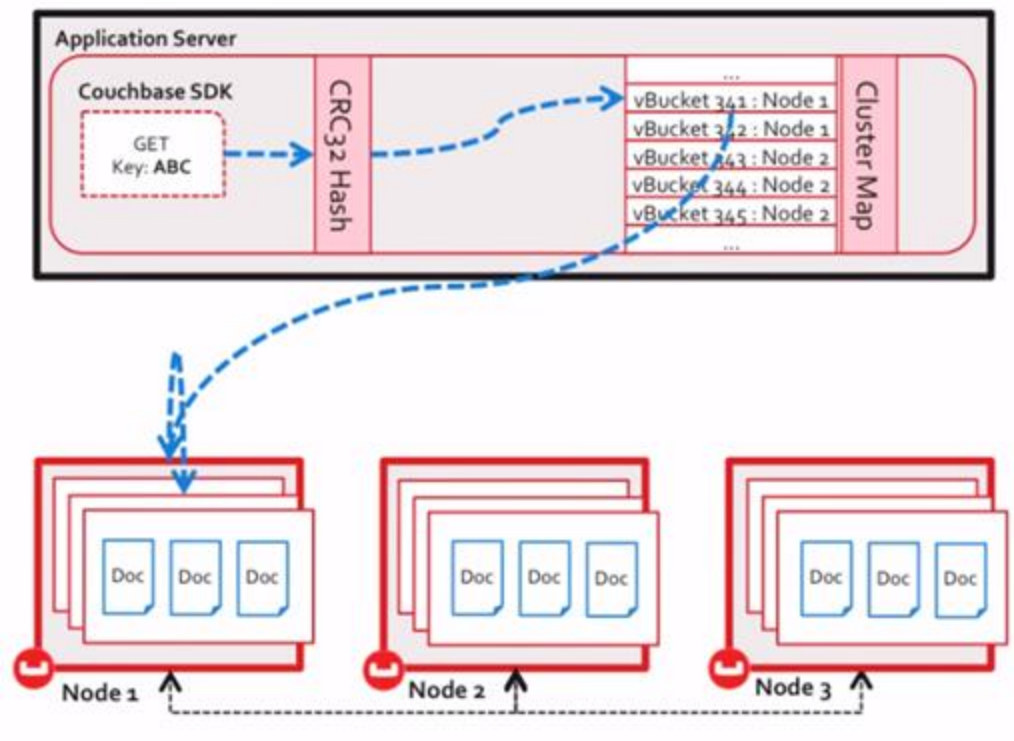
A document location is determined by its *key*

For any read or write, the key is run through a ***CRC32*** *hashing algorithm*

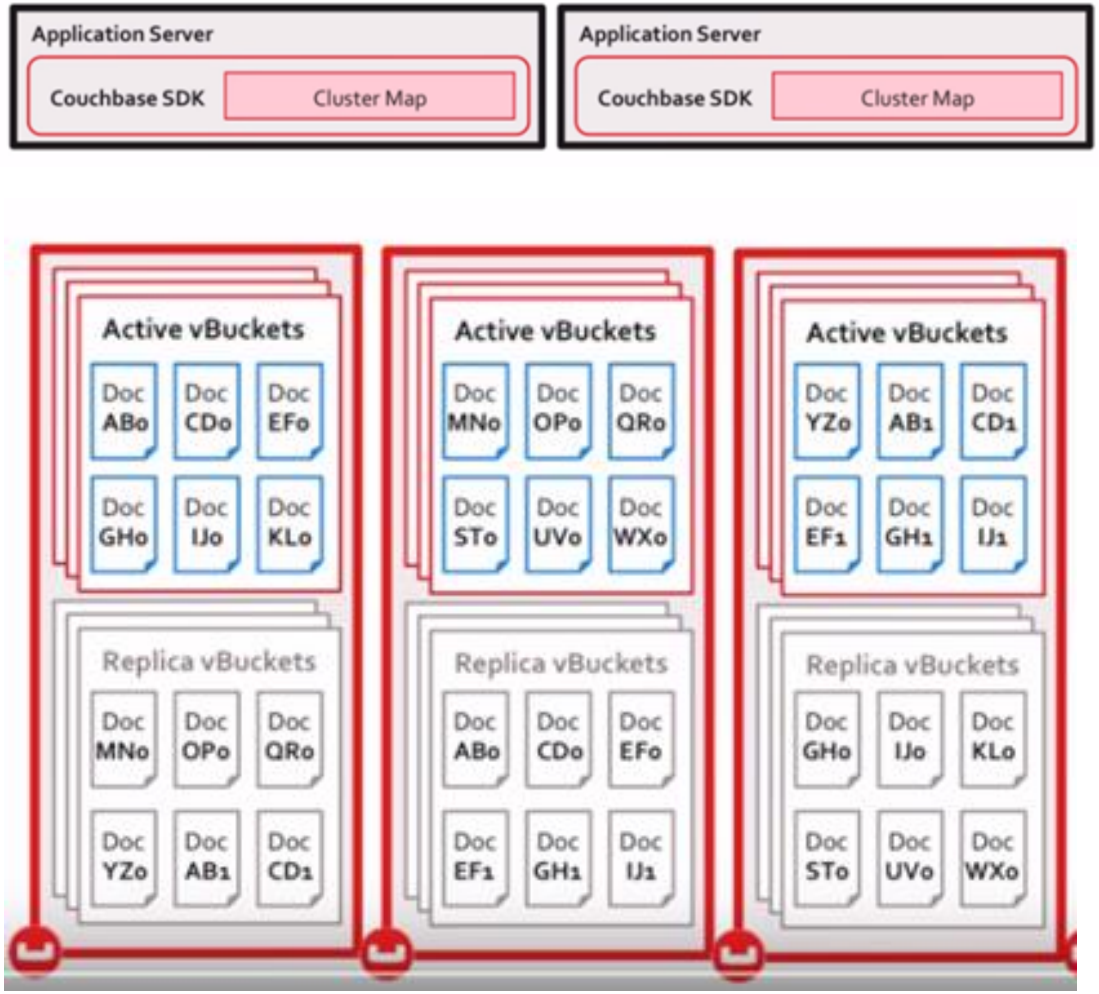
Hashed keys are distributed evenly across vBuckets, which are tracked in the **Cluster Map** of client's SDK

Cluster Map identifies the correct location for this read or write

- ✓ Node
- ✓ vBucket

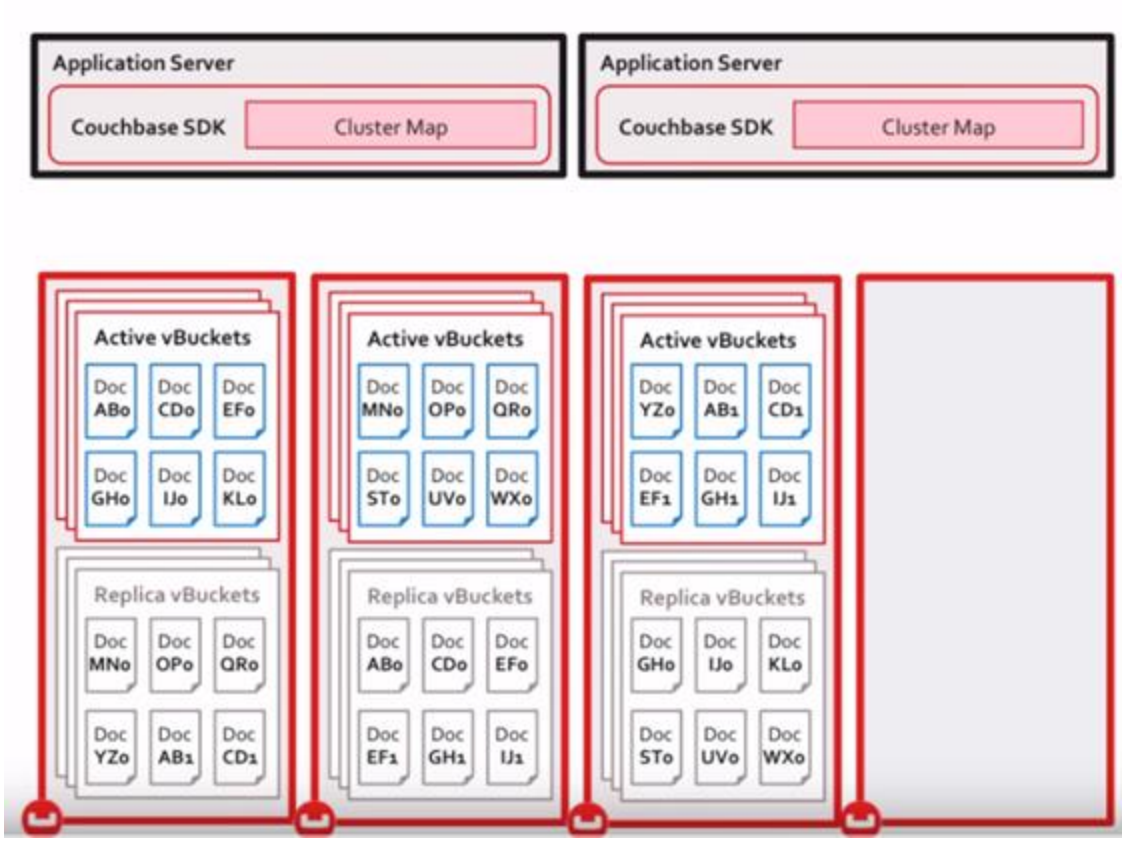


What about replication?



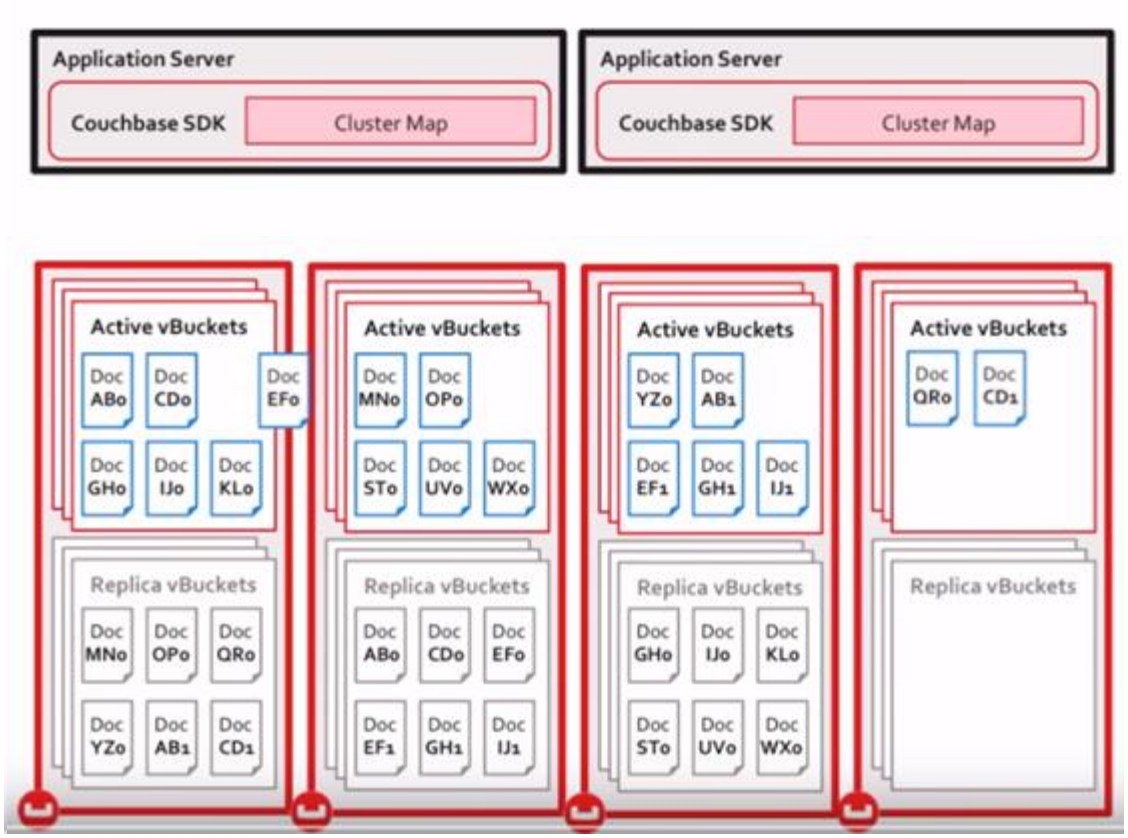
What happens when nodes are added to a cluster?

- ✓ New node address added via UI or REST



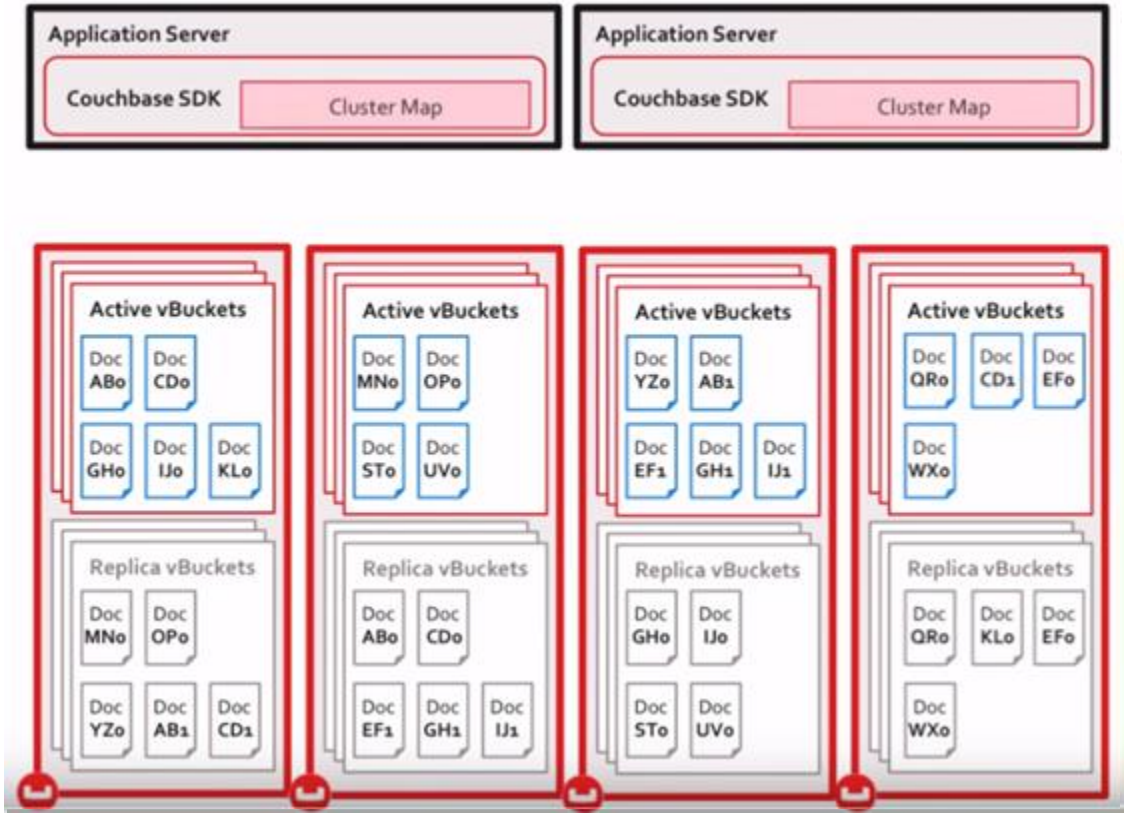
What happens when nodes are added to a cluster?

- ✓ New node address added via UI or REST
- ✓ vBuckets are recalculated for each Bucket
- ✓ Documents are incrementally transferred



What happens when nodes are added to a cluster?

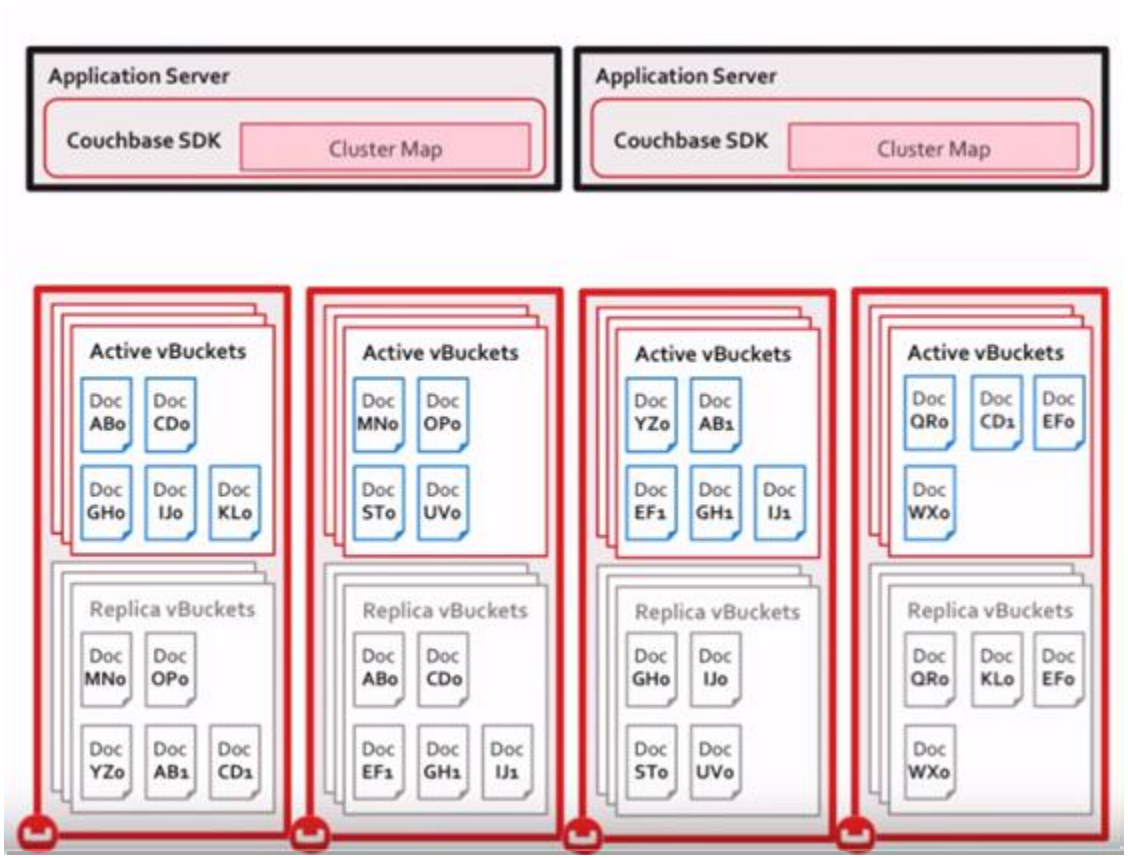
- ✓ New node address added via UI or REST
- ✓ vBuckets are recalculated for each Bucket
- ✓ Documents are incrementally transferred
- ✓ Updated cluster maps are continuously provided



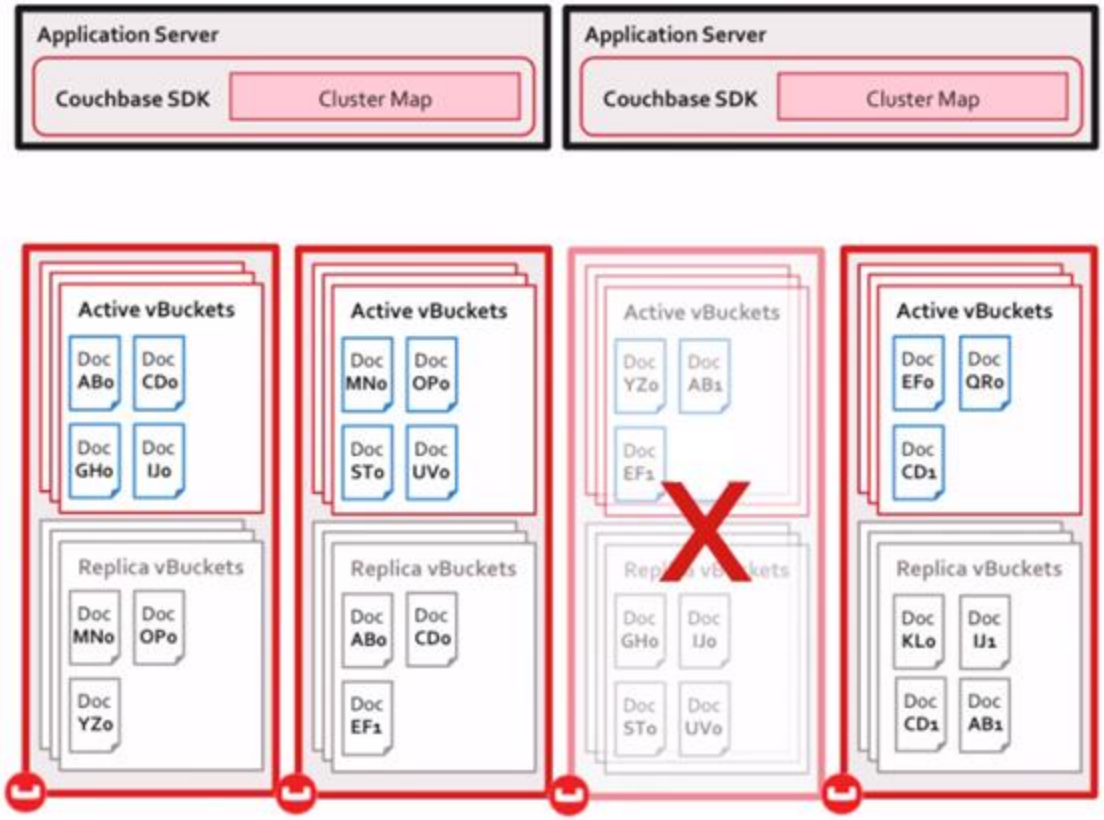
What happens when nodes are added to a cluster?

- ✓ New node address added via UI or REST
- ✓ vBuckets are recalculated for each Bucket
- ✓ Documents are incrementally transferred
- ✓ Updated cluster maps are continuously provided

Zero downtime

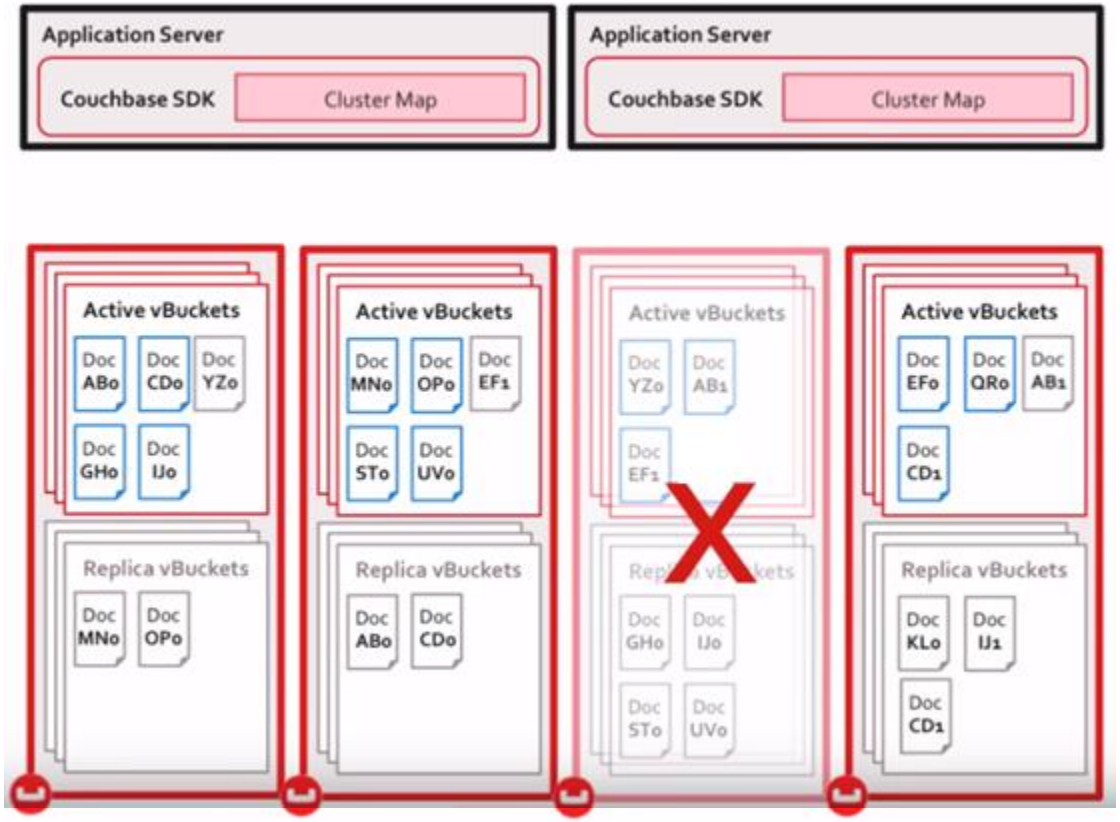


What happens when nodes are removed?



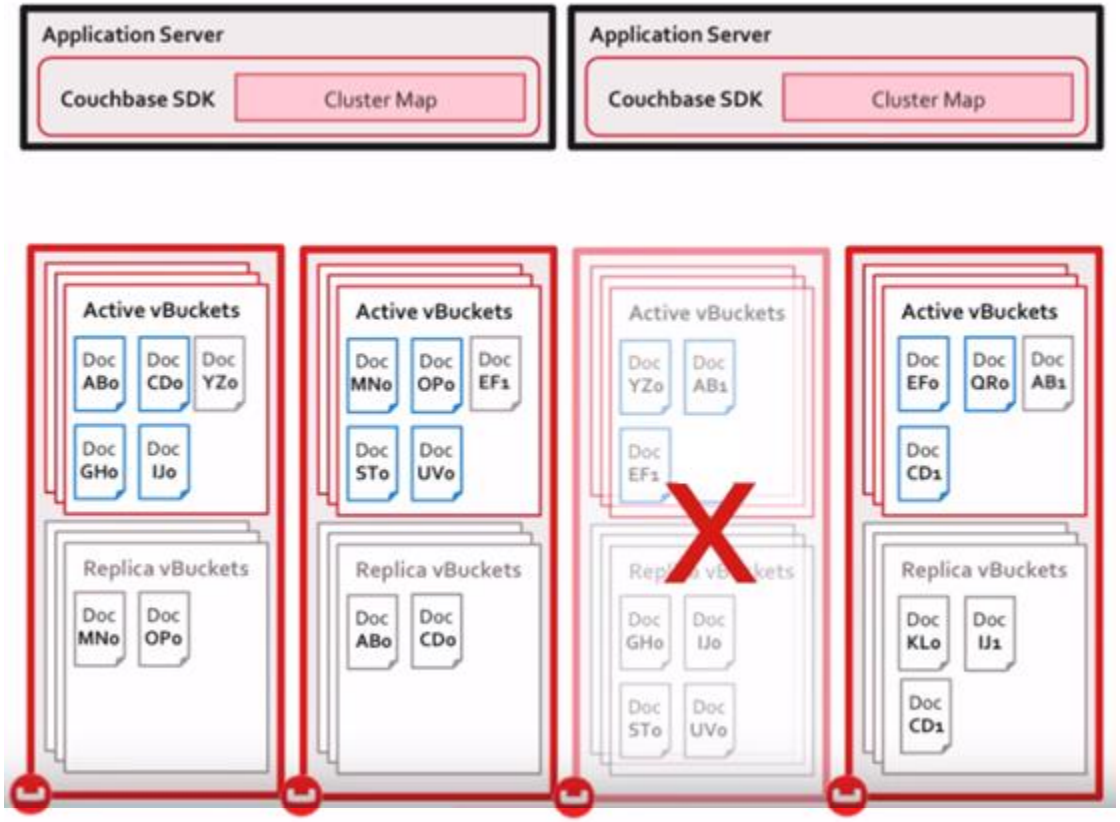
What happens when nodes are removed?

- ✓ Replicas promoted
- ✓ Cluster map updated



What happens when nodes are removed?

- ✓ Replicas promoted
- ✓ Cluster map updated



Zero downtime

What is XDCR?
(Cross Data Center Replication)

Secure, continuous memory-to-memory replication among clusters

Configured per bucket
SSL encrypted streams (default)
both intra-cluster and cross-cluster

Cluster topology neutral and aware

Each cluster may be differently sized and resourced
No loss auto-recovery if any node fails at either end

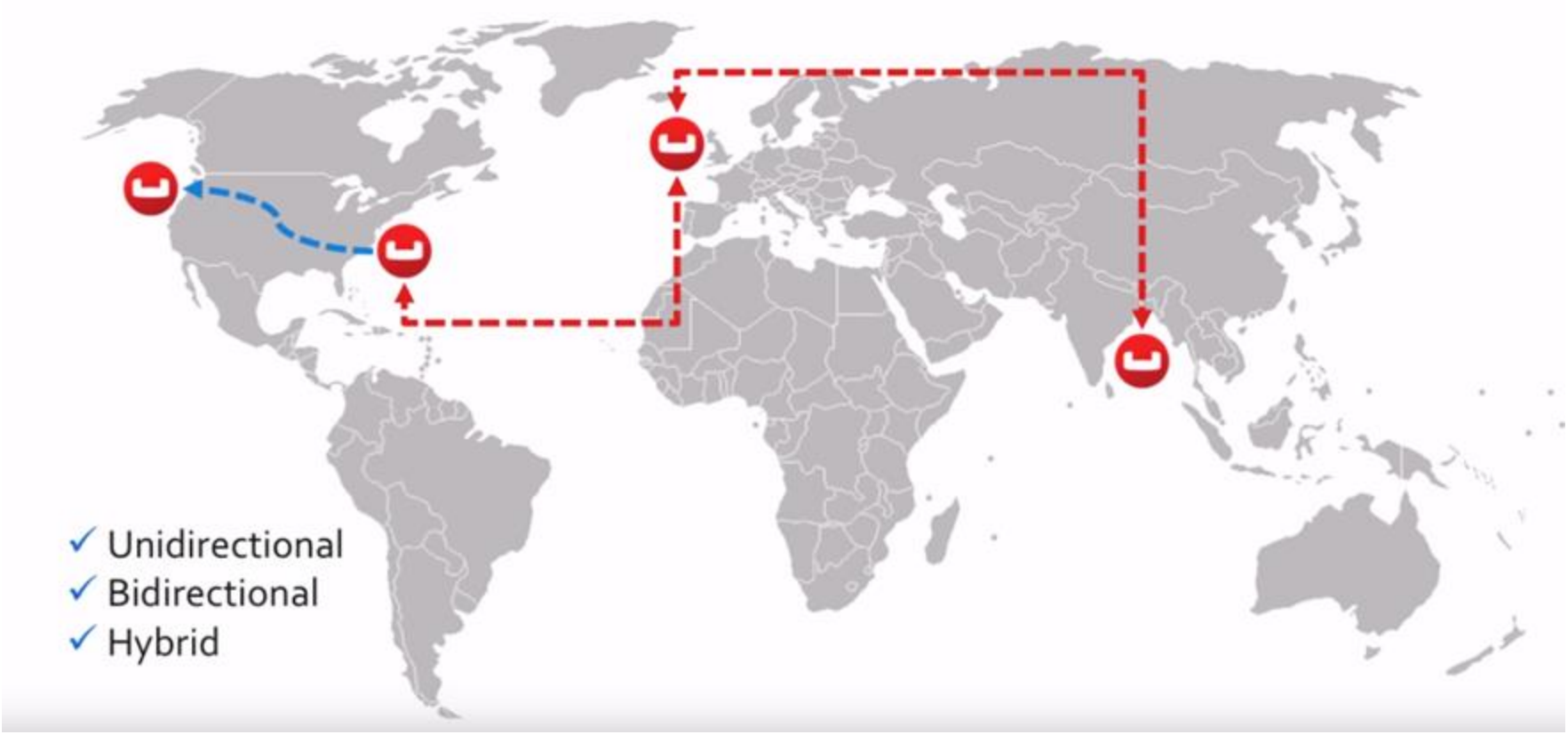
Efficient

When several mutations of a document are queued,
only the last is pushed remote

Resilient

Regular checkpoints to support pause/resume
Recoveries starts at most recent checkpoint

What topologies are available?



Couchbase SDKs & Tools

SDK manages connections, topology, documents, and queries

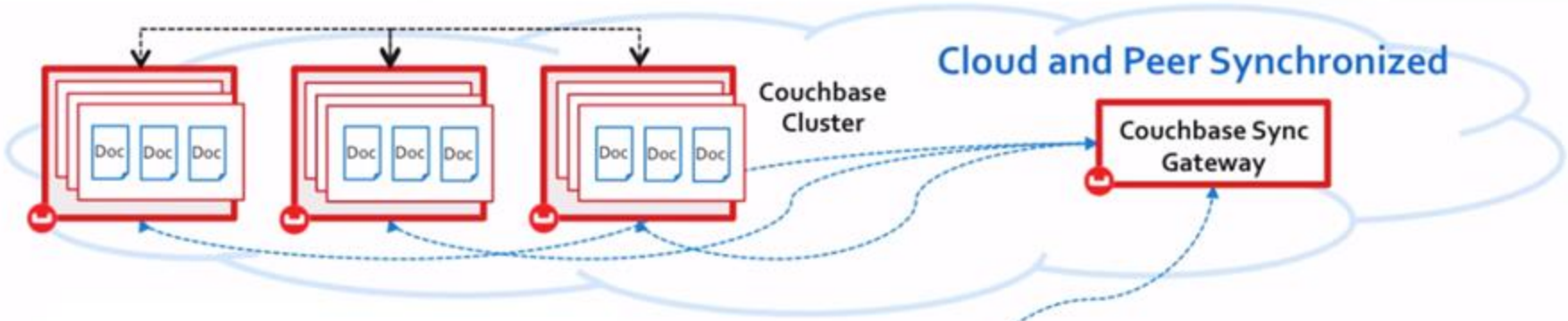
- ✓ Java
- ✓ C# / .Net
- ✓ Node.js
- ✓ PHP
- ✓ C/C++
- ✓ Python
- ✓ Ruby

Couchbase supports Reactive Programming

Couchbase is widely supported and easily integrated

- ✓ Analytical Databases: Apache Spark and Hadoop Connectors
- ✓ Business intelligence tools: ODBC/JDBC drivers
- ✓ Full text search: Apache Solr connector and Elasticsearch plugin
- ✓ Developer tools: Spring Data for Couchbase
- ✓ Big Data UI Integration: Talend connector for Couchbase
- ✓ Dozens more… Akka, Docker, Puppet, Ansible, New Relic, AppDynamics

How is Couchbase optimized for mobile development?

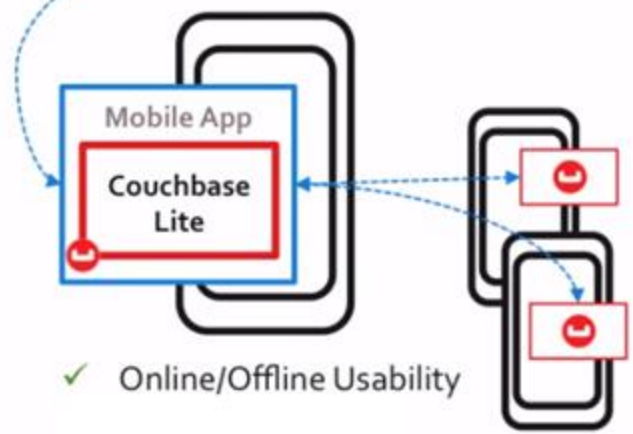


Couchbase Lite

- ✓ Embeded NoSQL database
- ✓ Lightweight and fully functional
- ✓ Native, on-device

Couchbase Sync Gateway

- ✓ Authentication and access control
- ✓ Replication and validation
- ✓ Data routing



Demo Time!

MAY THE DEMO GODS BE WITH US

References

- [1] “CB030 Essentials of Couchbase NoSQL Technology” [Online]. Available : https://training.couchbase.com/online