Couchbase
Learning Services

Agustín F. Calderón M.

Couchbase
Learning Services

CB030 Essentials of Couchbase NoSQL Technology

CB030 Essentials of Couchbase NoSQL Technology

Oct 23 , 2016

Couchbase

Learning Services

What is Couchbase?

Consistent. Elastic. Scalable. Always available.

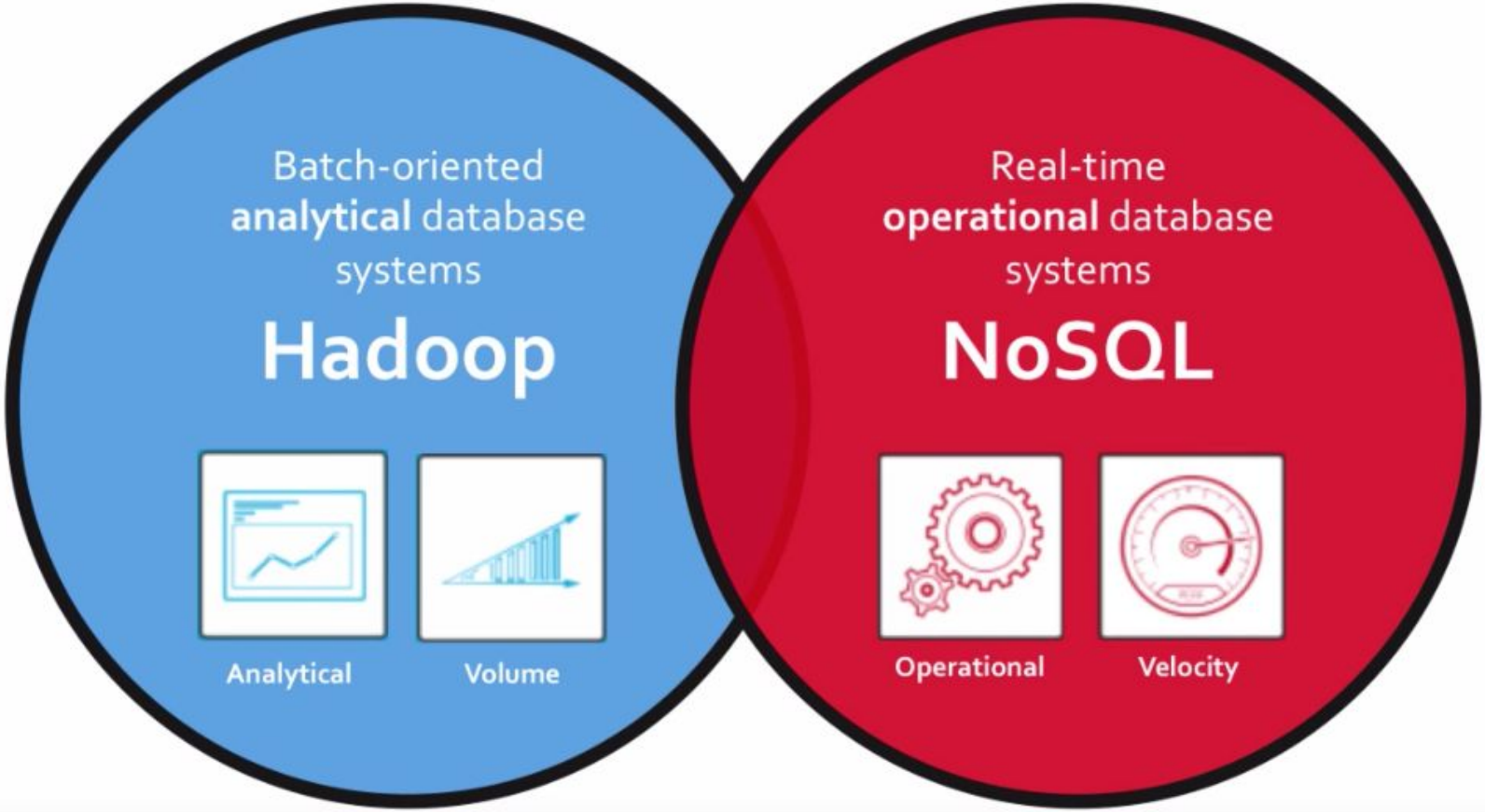
What is the project history?

Memcached
Distributed caching system

Membase
Distributed and persisted key-value store

Couchbase
Distributed and persisted document store with secondary indexing, querying with N1QL (SQL for JSON), and world-class tooling and integration

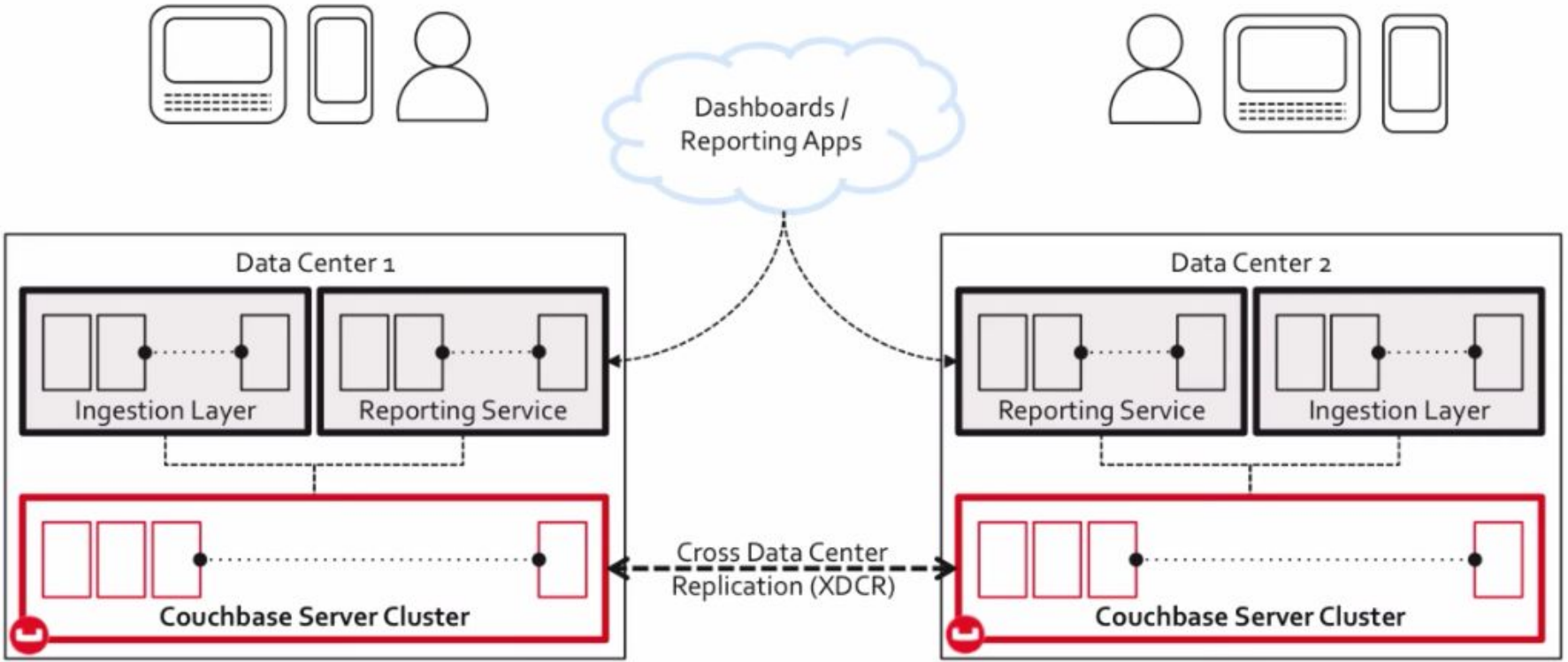
How are analytical and operational uses different?



How do you put all this data to work?

Analytical Use	Operational Use
Batched workloads	Real time intelligence
Vast data aggregations	Focus on data flows and processes
Retrospective analyses	Extremely fast (in-memory) reads
Focus on data pools	Extremely fast (log append) writes
Improve future outcomes	Improve the current outcome

Where would Couchbase fit into an enterprise app?



What does Couchbase provide?

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

High availability cache

Key-value store

Document database

Embedded database

Sync management

Enterprises often start with cache
then broaden usage to other apps and use cases

What we have?

Couchbase is the natural evolution beyond Memcached

Operational databases exist to provide real time intelligence

Couchbase is a complete operational data management solution

- ✓ High availability cache
- ✓ Key-value store
- ✓ Document database
- ✓ Mobile database
- ✓ Sync management

How is data organized?

Key-value pairs and documents.

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 storage location

Documents may be any binary value

- ✓ Null (key only)
- ✓ JSON encoded data, serialized object, XML, text, etc.

Each document includes metadata

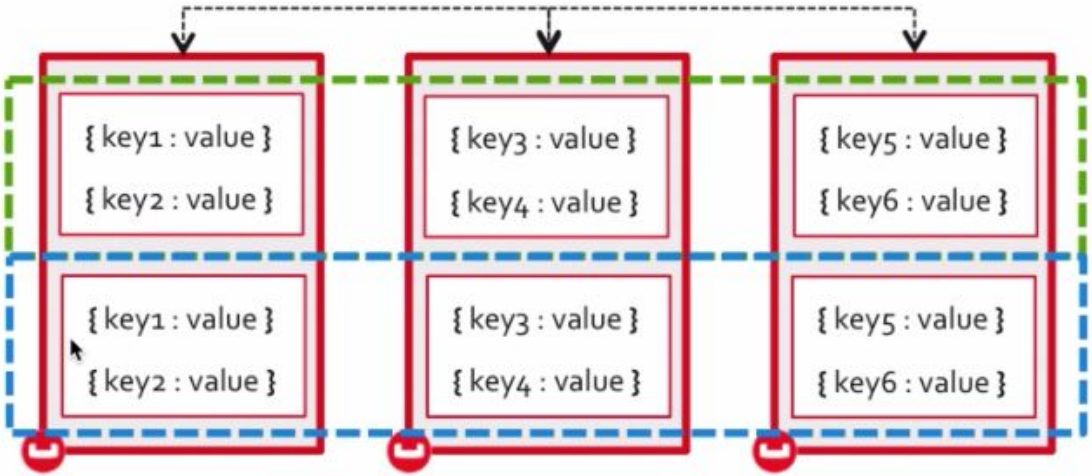
- ✓ Unique ID for optimistic concurrency (CAS)
- ✓ Optional expiration timestamp (TTL)
- ✓ Optional SDK specific flags (e.g., type, format)

What are the main architectural structures?

Node
A Couchbase Server instance

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

Bucket
A logical keyspace 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 three ways to get 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 based Views API

N1QL ("Nickel") Queries
- ✓ SQL superset for indexing and querying JSON documents
- ✓ Query service maintains *Global* secondary indexes
- ✓ Access either global or distributed indexes

What we have?

Couchbase stores documents, where each has

- ✓ Key – a document ID up 250 bytes, unique within a bucket
- ✓ Value – up to 20mb of binary data, usually JSON
- ✓ Metadata – values for concurrency, time to live, SDK flags

Couchbase organizes data among

- ✓ Documents – binary values with unique keys per *bucket*
- ✓ Buckets – document collections shared across *nodes*
- ✓ Nodes – instances of Couchbase Server in a *cluster*
- ✓ Cluster – a set of networked Couchbase nodes

Individual documents are accessed by key

Document sets are queried via N1QL or REST

What is JSON and how would you use it?

Flexible. Lightweight. Multi-dimensional.

What is JavaScript Object Notation?

Lightweight data interchange format based on JavaScript
All text, programming-language independent

Based on two structures

- ✓ Collections of key-value pairs wrapped in *braces* { }
- ✓ Ordered lists of values wrapped in *brackets* [ ]

Elements comma-separated ,
Text wrapped in *quotes* " "

```
{
  'first_name' : 'Roberta',
  'last_name' : 'Red',
  'age' : 37,
  'admin' : true,
  'logins' :
    [
      '2015-03-26-09:20AM',
      '2015-03-31-16:13PM'
    ],
  'pets' :
    [
      {'name' : 'Fido', 'type' : 'Chameleon'},
      {'name' : 'Rover', 'type' : 'Chicken'}
    ]
}
```

Want to see a few examples?

Array (ordered list) of strings

```
[ 'Jan', 'Piet', 'Yuki', 'Asya' ]
```

Object (collection) of primitives

```
{ 'name':'Jan', age:53, admin:true }
```

Array of objects

```
[
  { 'name':'Jan', age:53, admin:true },
  { 'name':'Piet', age:36, admin:false }
]
```

Array of objects with nested array

```
[
  { 'name':'Jan', age:53, admin:true,
    pets:['Fido', 'Rover']
  },
  { 'name':'Piet', age:36, admin:false,
    pets:['Tweety']
  },
]
```

Why does Couchbase focus on JSON?

JSON is easy to

- ✓ Store
- ✓ Retrieve
- ✓ Parse

JSON also

- ✓ Works in all coding languages
- ✓ Flexibly lets schema evolve
- ✓ Supports multiple dimensions
- ✓ Maps easily to objects

```
{'first_name':'Roberta','last_name':'Red',
'age':37,'admin':true,'logins':
['2015-03-26-09:20AM','2015-03-31-16:13PM'],
'pets':[{'name':'Fido','type':'Chameleon'},
{'name':'Rover','type':'Chicken'}]}
```

Why does Couchbase focus on JSON?

JSON is easy to

- ✓ Store
- ✓ Retrieve
- ✓ Parse

JSON also

- ✓ Works in all coding languages
- ✓ Flexibly lets schema evolve
- ✓ Supports multiple dimensions
- ✓ Maps easily to objects

Just drop it in a bucket, and retrieve it when you need it

Key	Value
user::**123**	{'first_name':'Roberta','last_name':'Red','age'37,'admin':true,'logins':['2015-03-26-09:20AM','2015-03-31-16:13PM'], 'pets':[{'name':'Fido','type':'Chameleon'},{'name':'Rover','type':'Chicken'}]}
order::**42**	{order:42, customer: {name:"BipCo", address:"23 5th", email:"us@bip"}, items: [{qnty:4, name:"Widget", desc:"Fit for all uses."}, {qnty:2, name:"Gadget", desc:"Just what I need."}], paytype: {name:"Visa"}}

Is JSON well supported?

JSON is universally supported

- ✓ 178 JSON parsing libs for
- ✓ 63 different languages

References and tools include

- ✓ json.org – resource lists
- ✓ jsonlint.com – validator
- ✓ jsoneditoronline.org – editor

Standardized in RFC 7159

Do you have to use JSON?

No.

Each Couchbase document
stores up to 20mb
of any binary value you like.

(serialized objects, XML, etc.)

What have we learned?

JSON is a very widely used data interchange format

- ✓ Standardized as IETF RFC 7159

Based on collections (objects) and ordered lists (arrays)

JSON.org lists 178 parsing libs across 63 languages

Relied on by Couchbase because

- ✓ Widely available and easily understood
- ✓ Allows schema to evolve
- ✓ Supports multiple dimensions
- ✓ Maps easily to objects

Couchbase documents can hold any binary value up to 20mb

How is a node organized?

Identically, but with three scalable services.

What is the high level architecture?

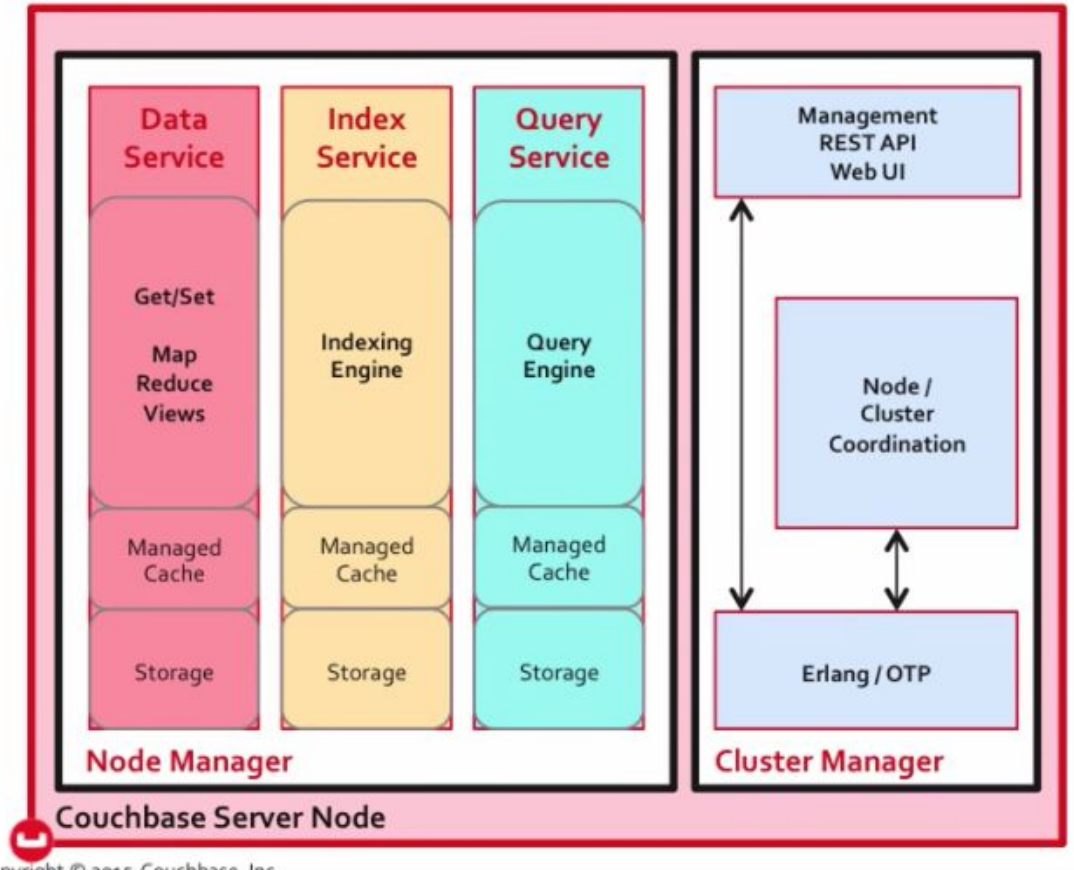
Couchbase Server nodes are identical

Two core components

- ✓ Cluster Manager
- ✓ Node Manager

Three independent services

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



What does the Cluster Manager Provide?

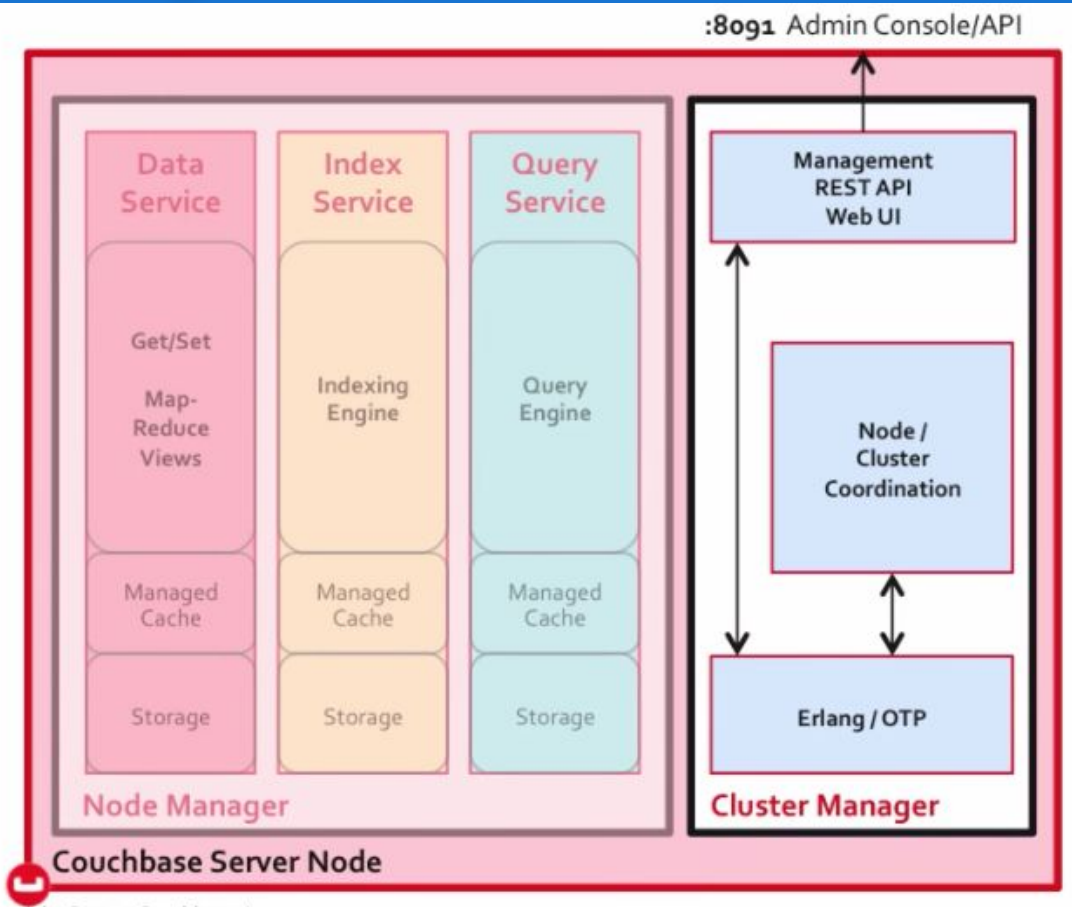
Management and coordination

Responsible for

- ✓ UI / REST interface
- ✓ Configuration / Administration
- ✓ Coordinating cluster rebalancing
- ✓ Process monitoring
- ✓ Statistics
- ✓ No data processing

Erlang/OTP code base

- ✓ Developed by Ericsson
- ✓ Telco-switch grade performance



What does the Node Manager provide?

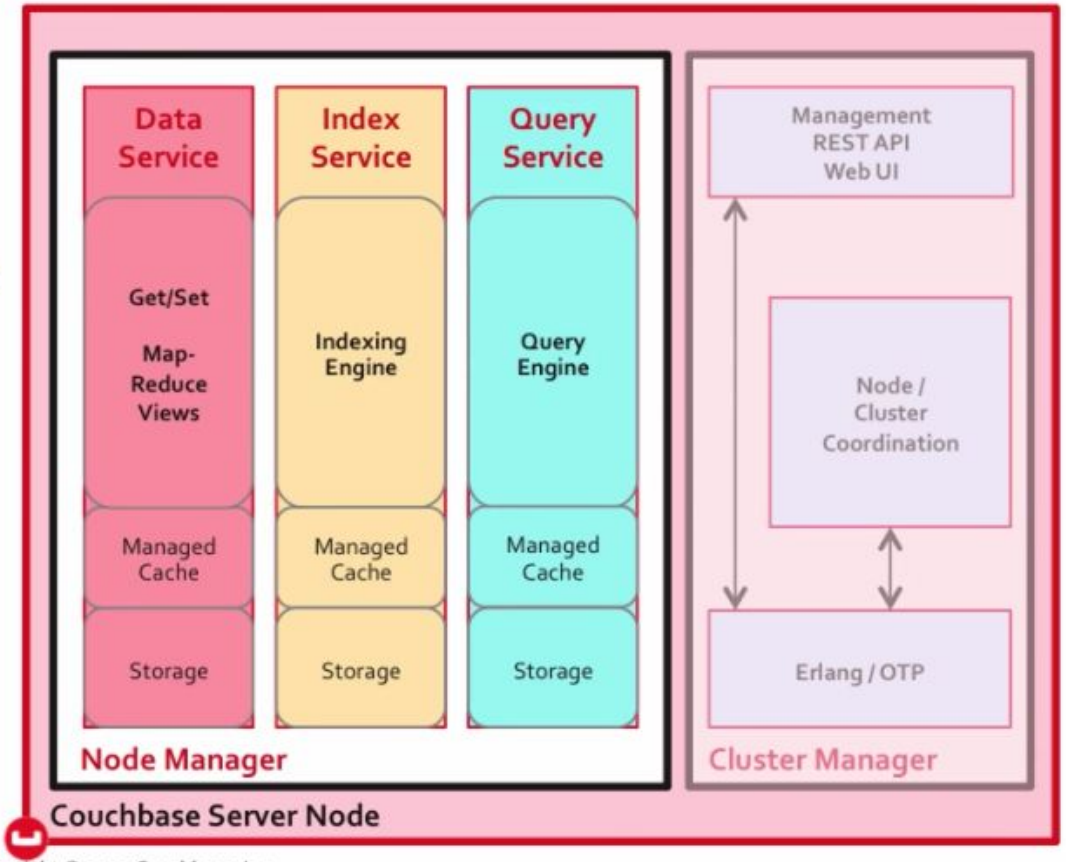
Three supported workloads

Data Service
- ✓ Get/Set of individual documents
- ✓ Distributed secondary indexes (Views)

Index Service
- ✓ Global secondary indexes
- ✓ Index maintenance and provision

Query Service
- ✓ N1QL (SQL superset for JSON)
- ✓ Execution plans, index coordination

C / C++ and Go code bases
- ✓ Very fast and memory efficient even with large memory footprint



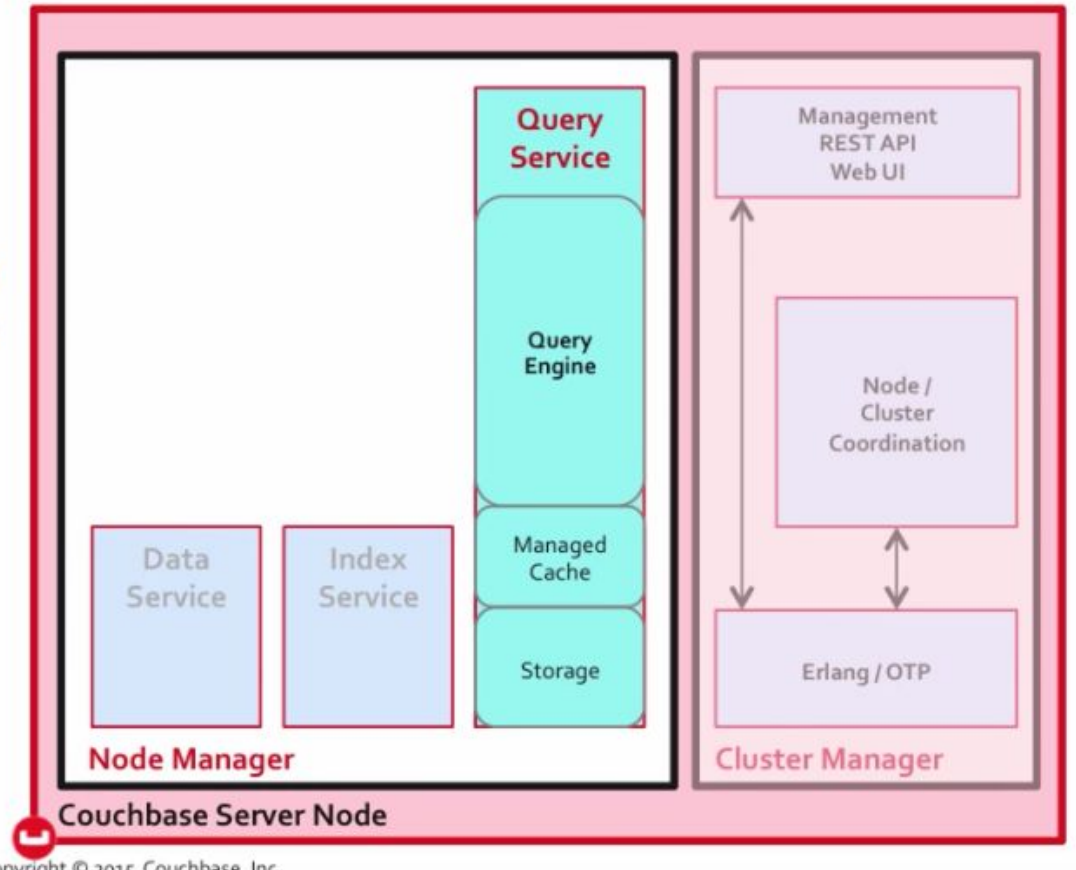
What is multi-dimensional scalability (MDS)?

Single node type / code base

- ✓ All services available on every node
- ✓ Services enabled independently

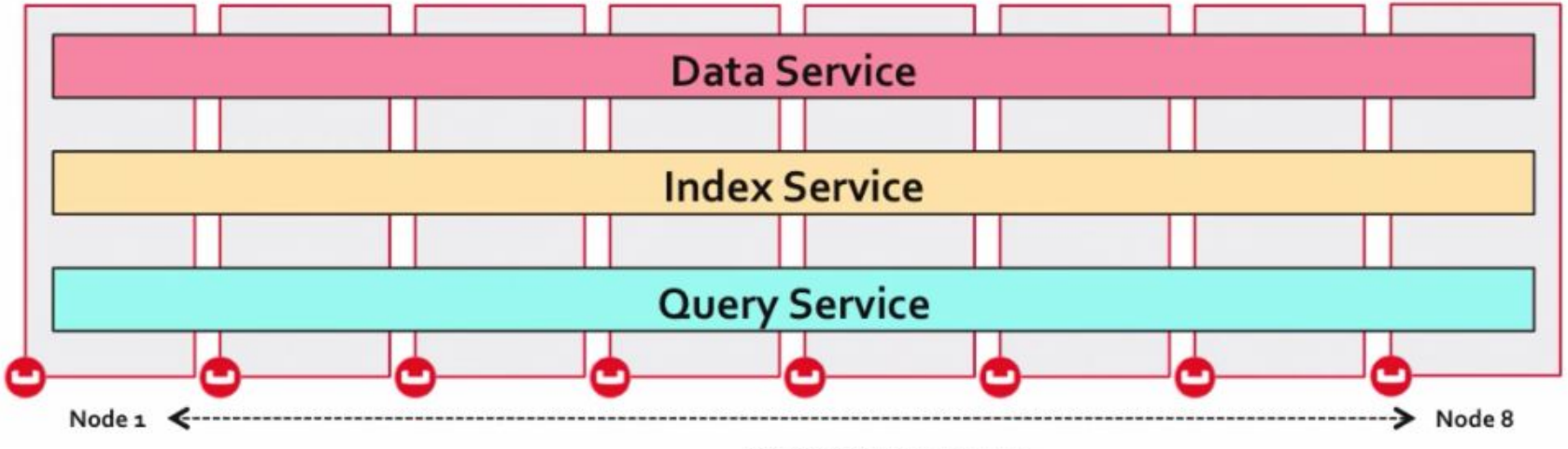
Allows cluster to be configured for

- ✓ Varying hardware capacities per node
- ✓ Application workload tuning



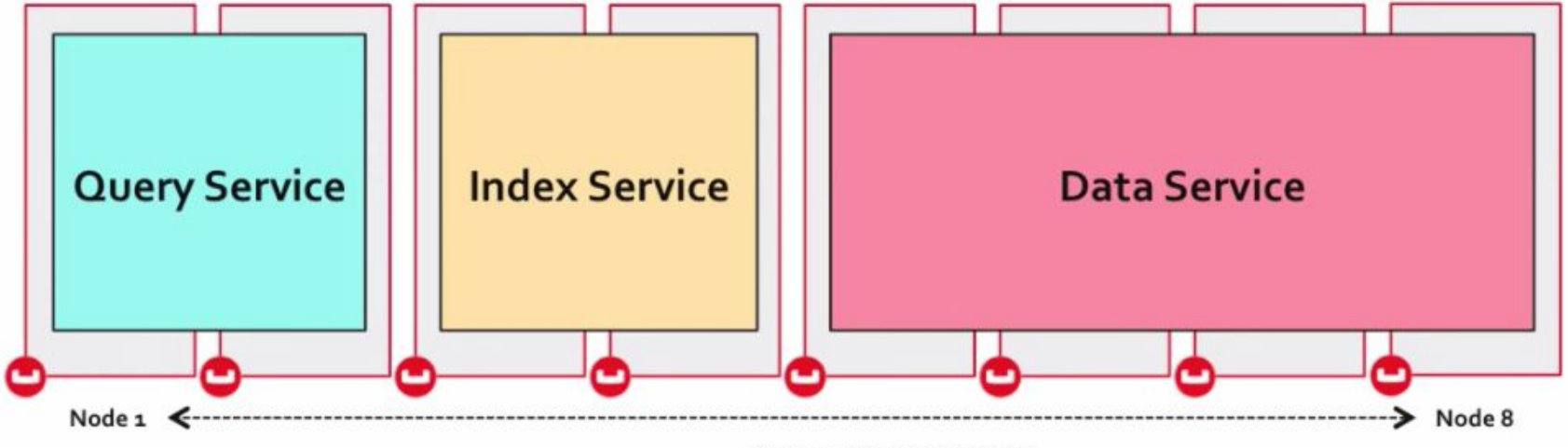
How can services be scaled throughout a cluster?

1. Stripe equally across all nodes



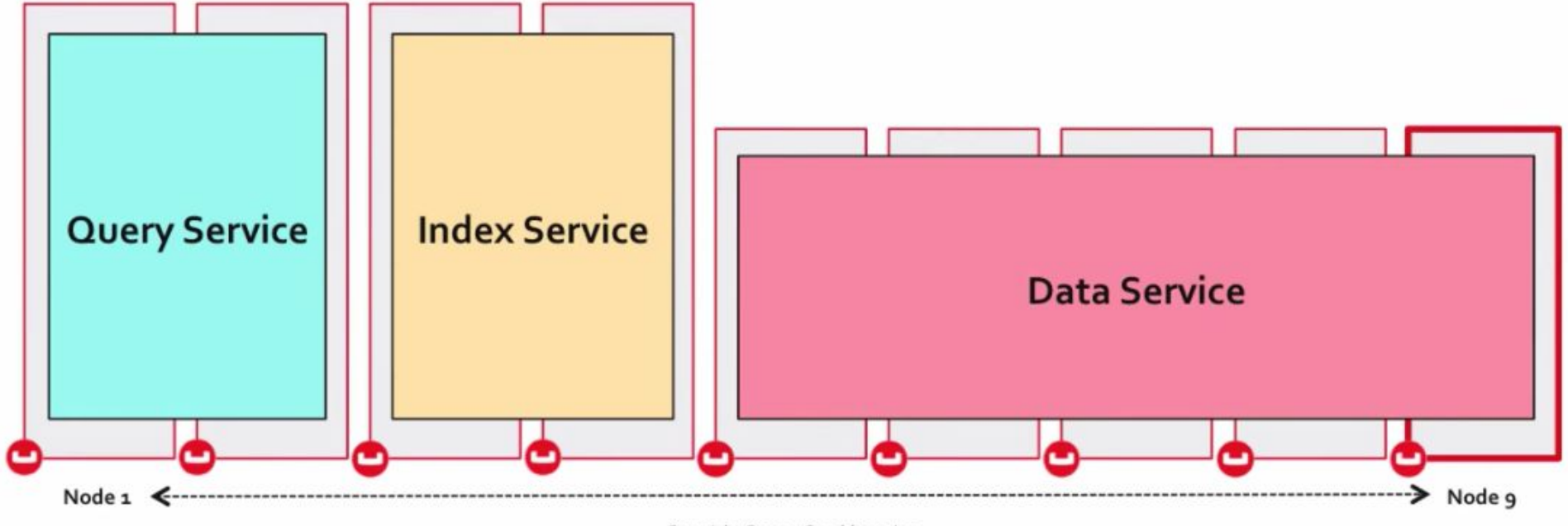
How can services be scaled throughout a cluster?

1. Stripe equally across all nodes
2. Isolate by service to minimize interference



How can services be scaled throughout a cluster?

1. Stripe equally across all nodes
2. Isolate by service to minimize interference
3. Scale independently for best hardware by service type



What we have?

Couchbase Server nodes are identical, and provide

- ✓ Cluster Manager – telco-switch grade manager and coordinator
- ✓ Node Manager – supports workloads through three services

Data Service

- ✓ Get/Set for individual documents
- ✓ MapReduce Views (secondary indexes) via API or N1QL

Index Service

- ✓ Maintain and support Global Secondary Indexes

Query Service

- ✓ Coordinate and execute N1QL queries (SQL for Documents)

Services may be evenly distributed or individually enabled

How are specific documents written and read?

Very, very quickly. It’s memory first.

Show me a bit of code ...

Connect, create, and retrieve a document (Java)

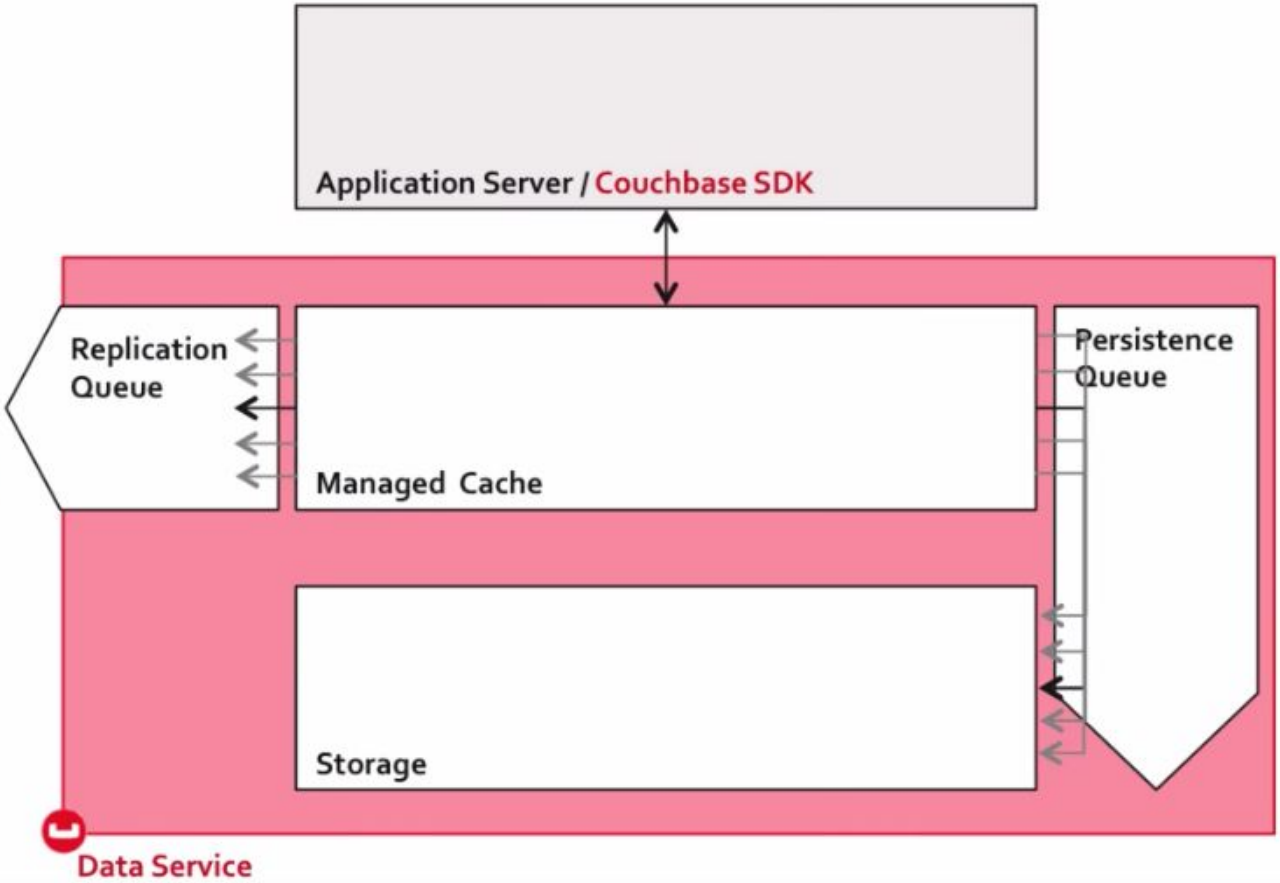
- ✓ Create connection
- ✓ Open bucket
- ✓ Create JSON object
- ✓ Create doc from object, assign key, and "upsert"
  - if exists, **up**date
  - if not, in**sert**
- ✓ Get the document by key, and display

```
// Connect to localhost
Cluster cluster = CouchbaseCluster.create("127.0.0.1");

// Open a bucket connection
Bucket bucket = cluster.openBucket("customers");

// Create a document
JsonObject user = JsonObject.empty()
  .put("first", "Walter")
  .put("last", "White")
  .put("job", "chemistry teacher")
  .put("age", 50);

// Store a document
JsonDocument stored =
    bucket.upsert(JsonDocument.create("walter", user));

// Get the document
JsonDocument walter = bucket.get("walter");
System.out.println("Found: " + walter.getString("first"));
```

What are the main Data Service components?

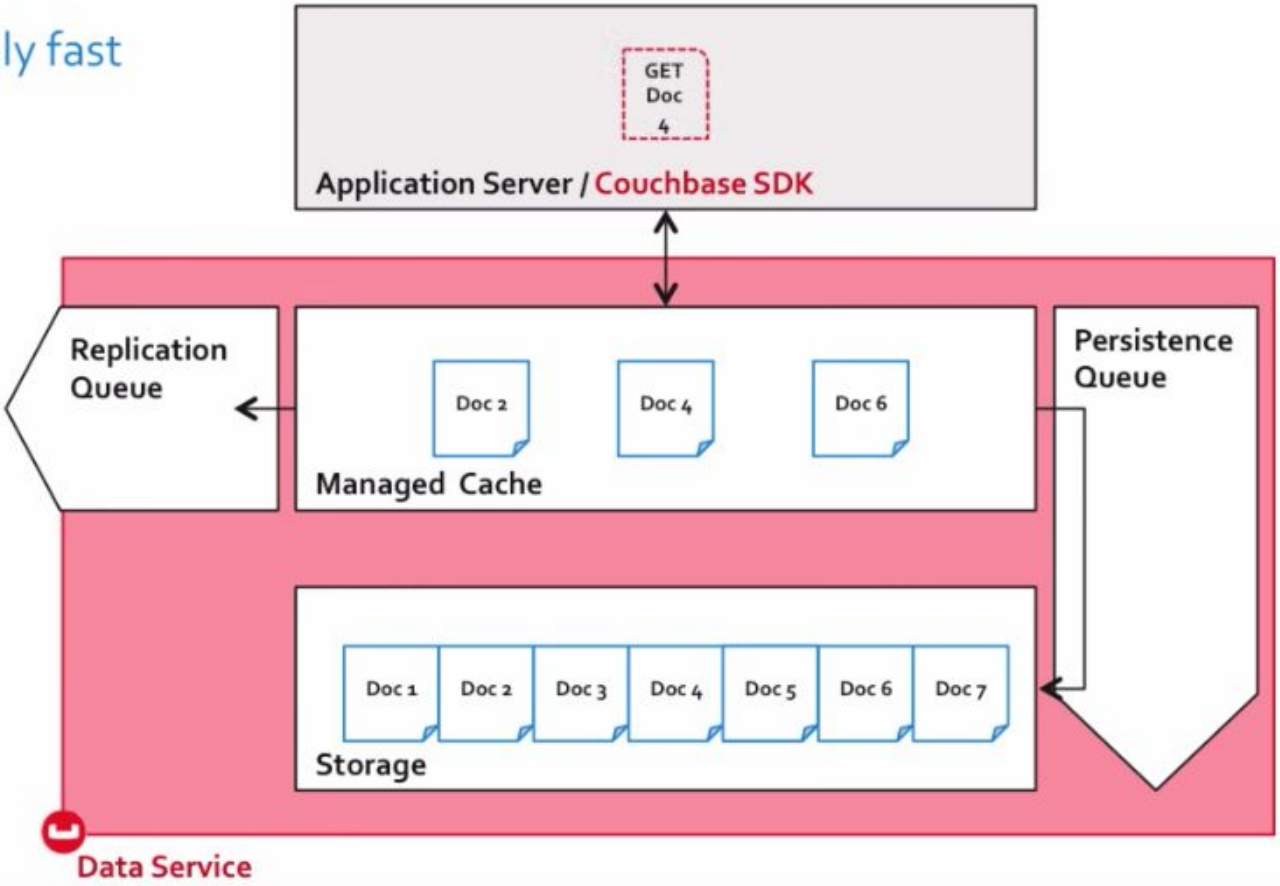
Five key components
- ✓ Managed Cache
- ✓ Persistence Queue
- ✓ Storage
- ✓ Replication Queue
- ✓ Application Server

All queues rely on shared, multi-thread pools
- ✓ Replication queue
- ✓ Persistence queue



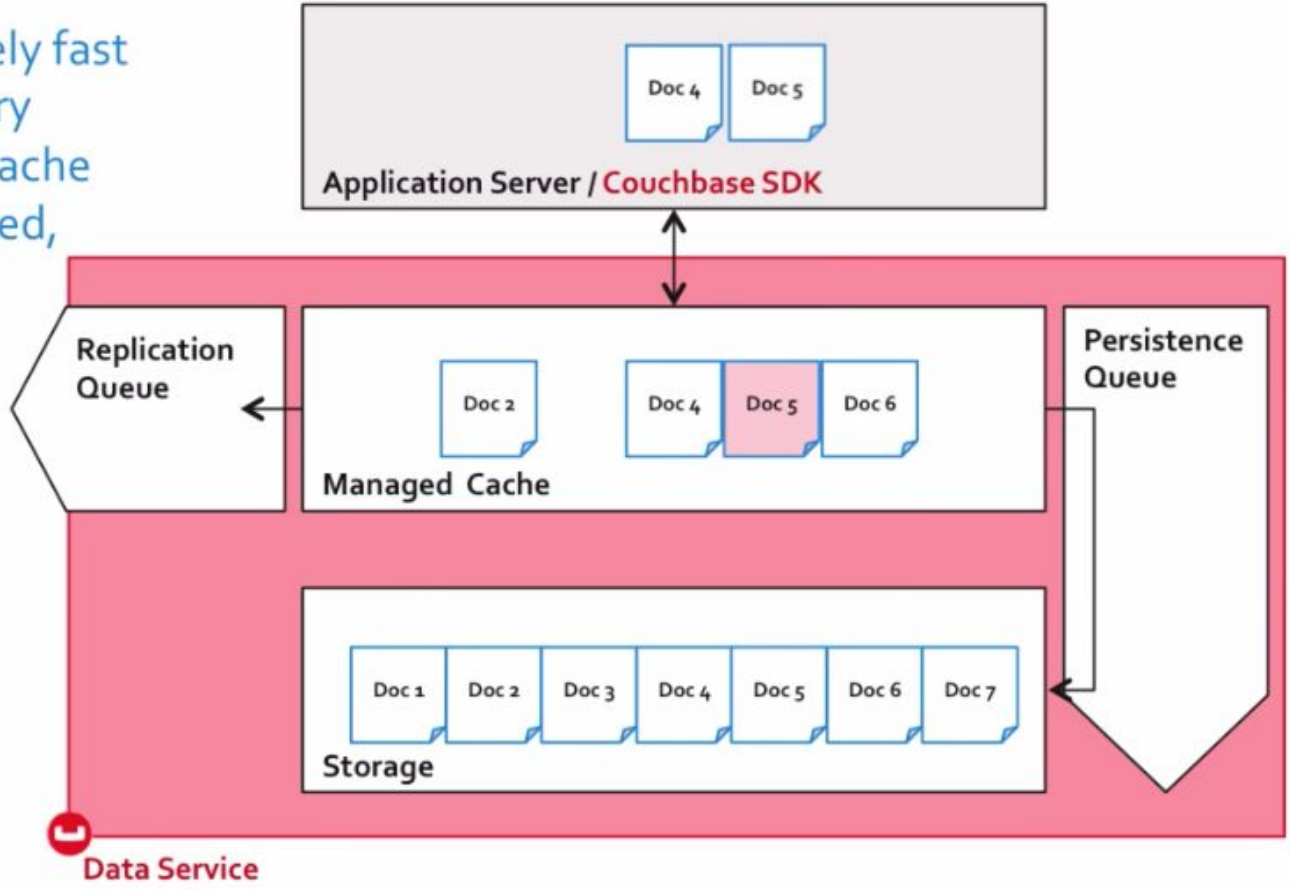
How does a get operate?

✓ Cache gets are extremely fast



How does a get operate?

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

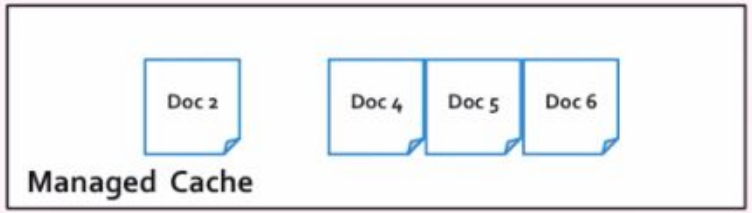


How is the cache managed?

NRU (Not Recently Used) score maintained for each cache item

- ✓ Each read lowers score
- ✓ Periodic *item pager* raises score

If memory use hits *high watermark*, eject high score items until *low watermark* achieved



Data Service

How is memory tunable by cache ejection type?

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

- ✓ **Value-only ejection** – key and metadata remain, only value removed
- ✓ **Full ejection** – key, metadata, and value all removed

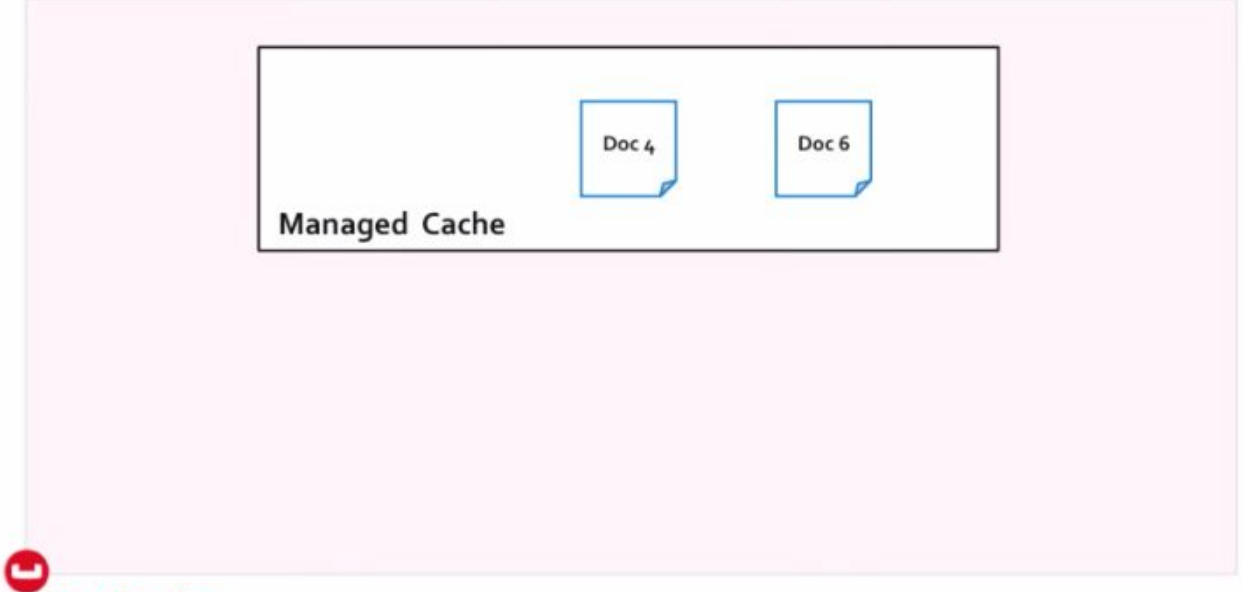
Value-only

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

Full ejection

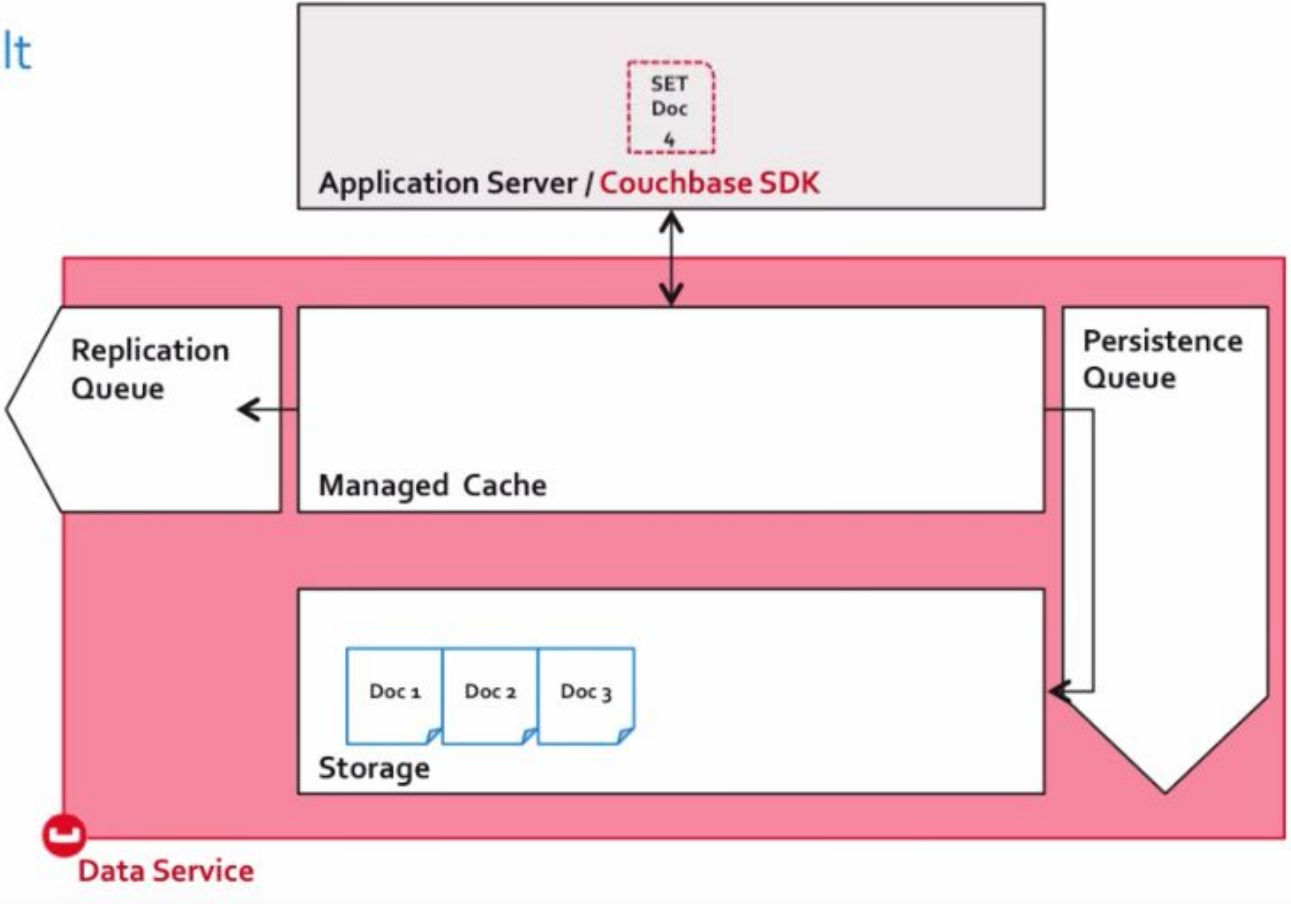
- ✓ Slower lookup speed
- ✓ Lower memory use

Best choice varies by use case



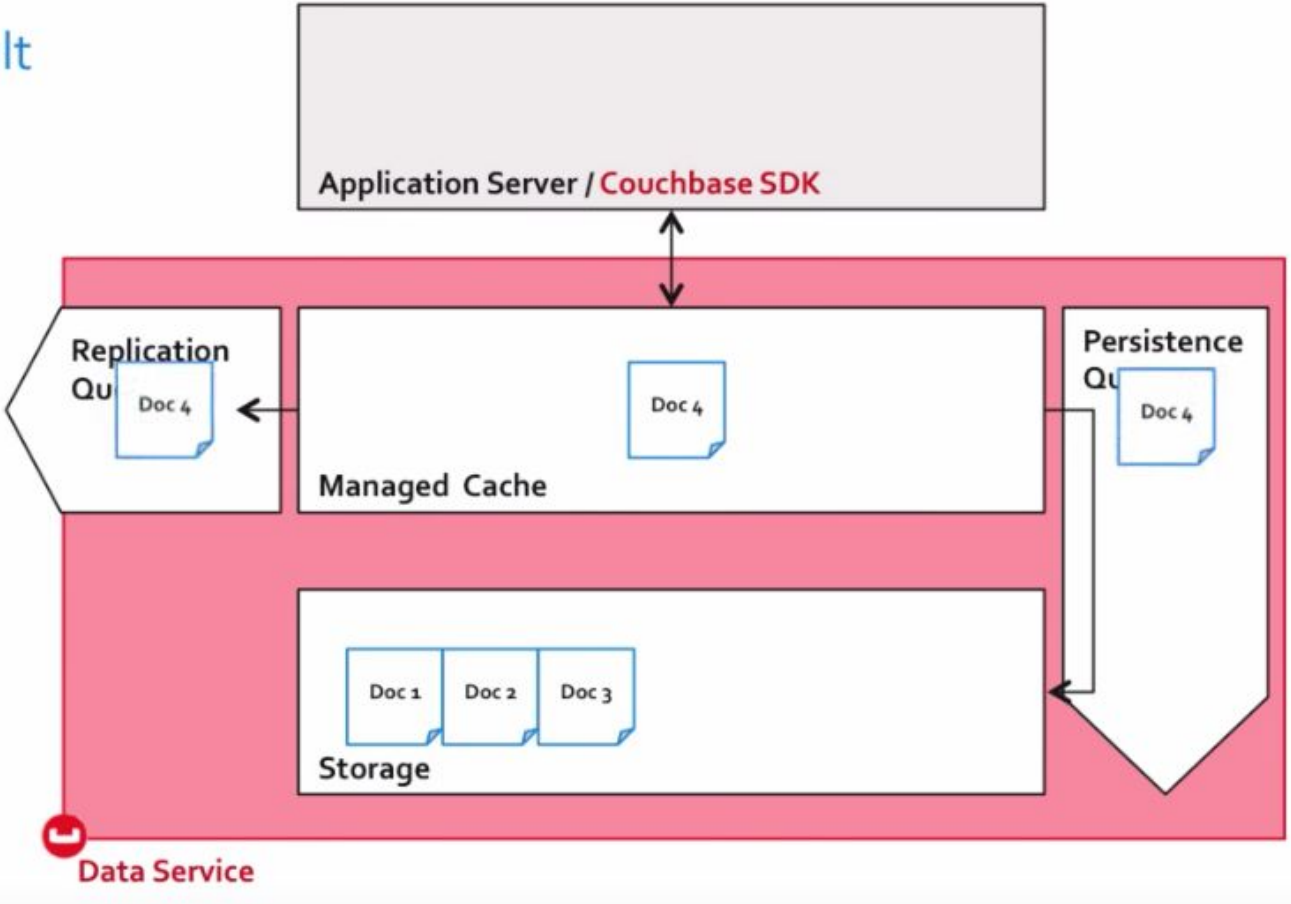
How does a set operate?

Sets are async by default



How does a set operate?

Sets are async by default



How does a set operate?

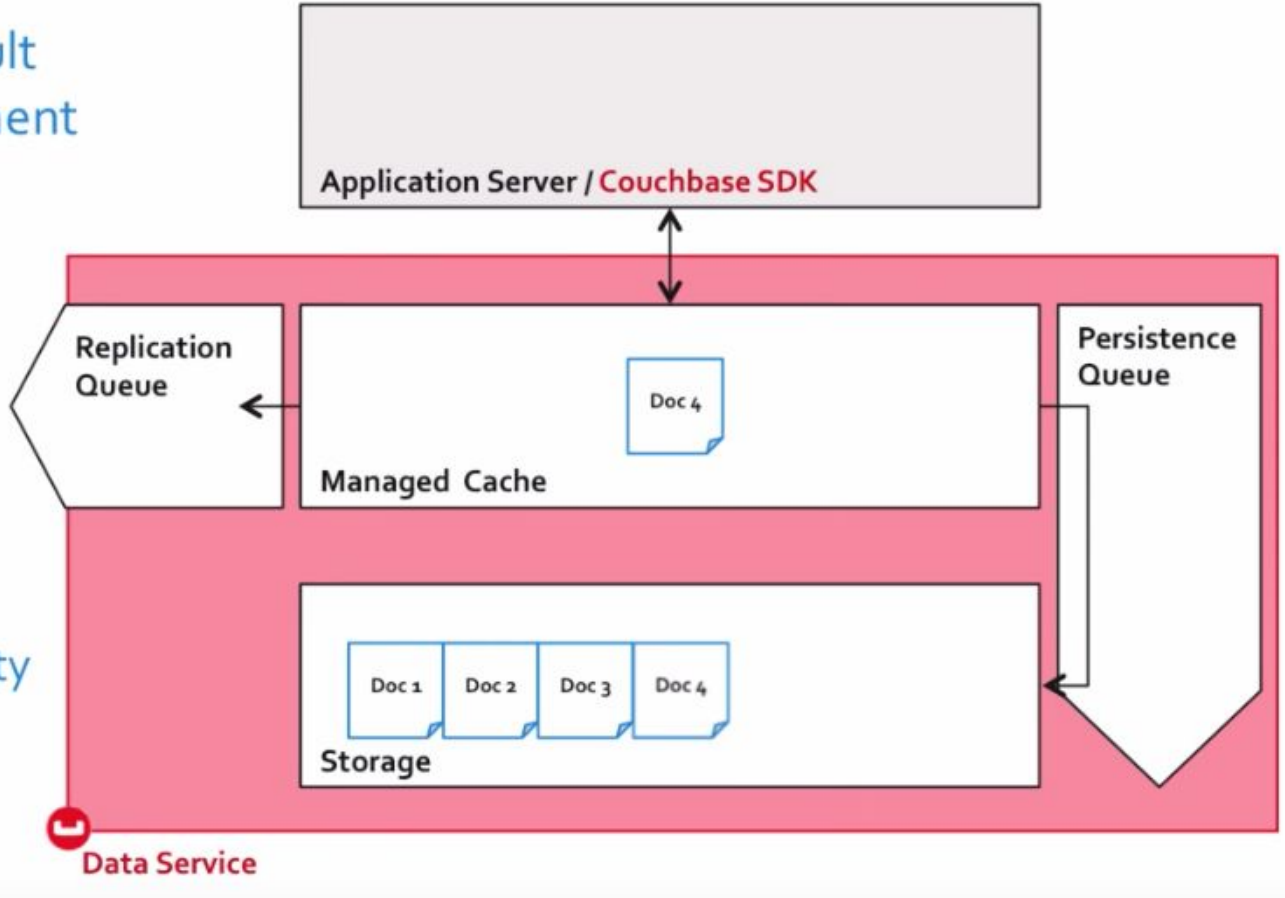
Sets are async by default

Choose acknowledgement type *per write*

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

Replication

- ✓ 1, 2, or 3 more nodes
- ✓ very fast, RAM to RAM
- ✓ off-node replication provides high availability
- ✓ nothing waits, unless you choose



Why is Couchbase Server so fast?

Gets and sets are served primarily from cache
Disk writes are append-only

Why is Couchbase Server so fast?

Gets and sets are served primarily from cache

Disk writes are append-only

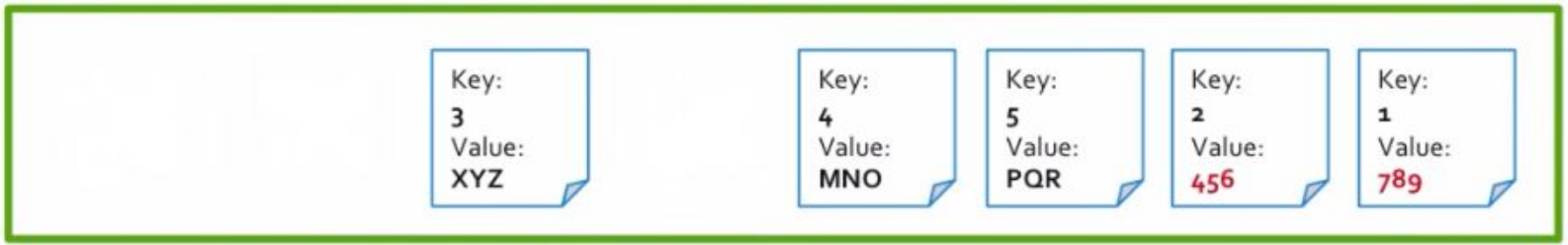
Why is Couchbase Server so fast?

Gets and sets are served primarily from cache

Disk writes are append-only

Data files are periodically compacted

✓ Tombstoned documents evicted



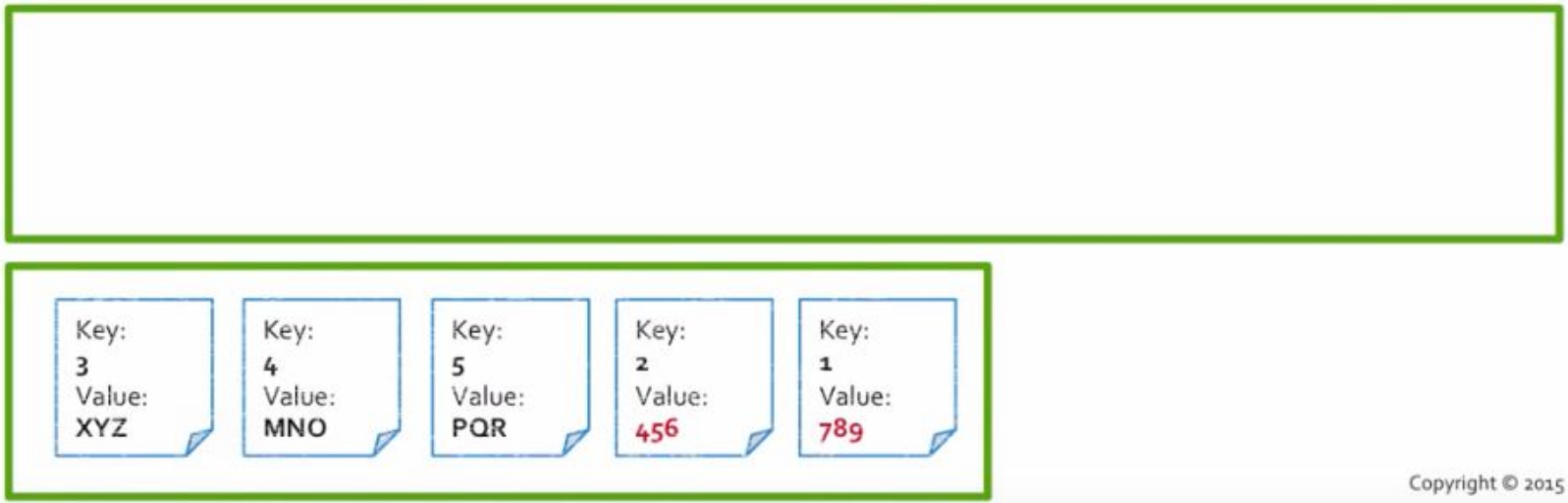
Why is Couchbase Server so fast?

Gets and sets are served primarily from cache

Disk writes are append-only

Data files are periodically compacted

- ✓ Tombstoned documents evicted
- ✓ Compacted replacement created and put online



Why is Couchbase Server so fast?

Gets and sets are served primarily from cache

Disk writes are append-only

Data files are periodically compacted

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

What we have?

Get/Set operations rely on Managed cache, Persistence queue, Replication queue, Storage, Couchbase SDK

Client connection is TCP binary

Due to working set management, most gets served from cache

- ✓ Common documents remain in cache
- ✓ Configurable whether metadata remains in cache

Sets are acknowledged when in RAM by default

May also acknowledge on persistence or replication, per set

Replication is RAM to RAM

Disk writes are append-only, and so periodically compacted

What if you need a set of documents, or to seek by a different key?

SQL? For Documents? But, you said “No SQL” …?

How else can you access documents?

Secondary Indexing

Get documents by a different key than their original key

Querying

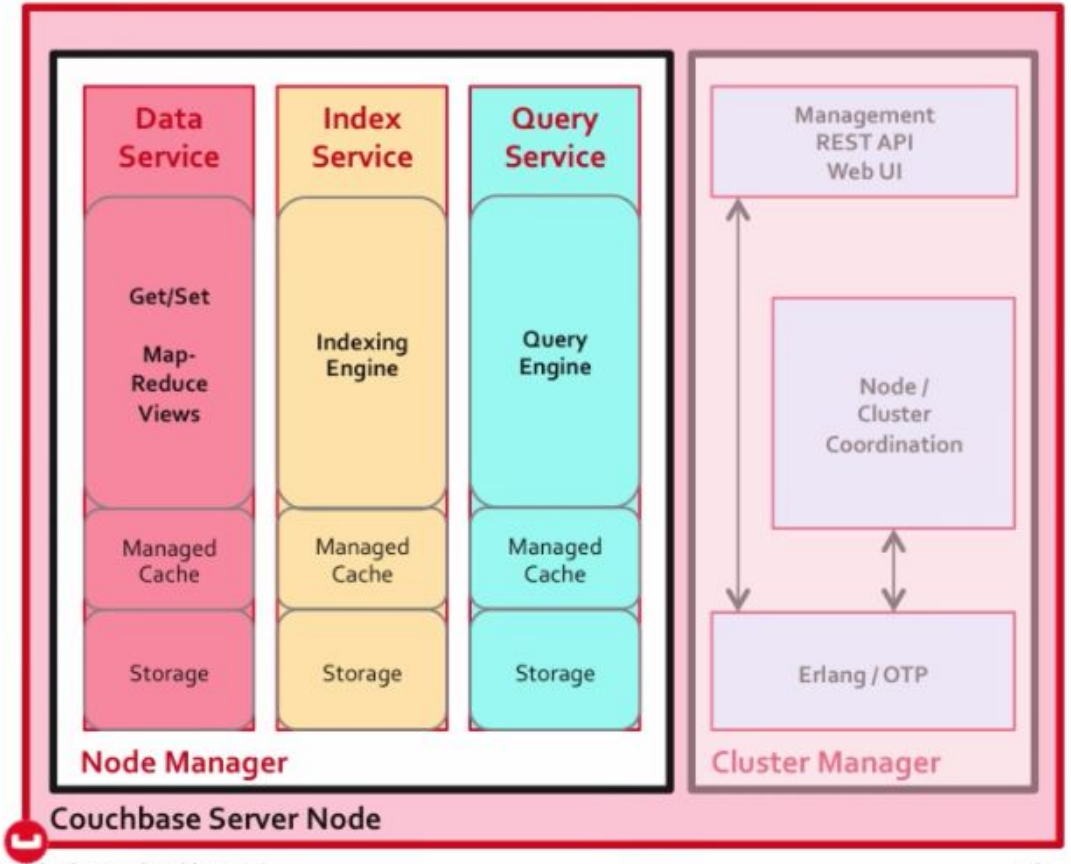
Create, read, update, or delete documents
where specific conditions match

Aggregation

Count, sum, evaluate, or otherwise gather collective data
where documents match specific conditions

How does Couchbase provide these capabilities?

- ✓ **Data Service** – builds and maintains Distributed secondary indexes (MapReduce Views)
- ✓ **Indexing Engine** – builds and maintains Global Secondary Indexes
- ✓ **Query Engine** – plans, coordinates, and executes queries against either Global or Distributed indexes



How does Couchbase provide these capabilities?

N1QL ("SQL for Documents")
Couchbase 4.0+

Superset of Structured Query Language
Nested, multi-dimensional data
Heterogeneous, semi-structured data
Distributed data, partitioned over clusters

Simplified code
Ad hoc queries
ODBC/JDBC for visualization and analysis

MapReduce Views
Couchbase 2.0+

Custom map functions define new keys
Built-in or custom reduce functions
Sum, count, or statistics

REST API for querying View-indexed data
N1QL queries can also access these Views

How do MapReduce Views work?

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

- ✓Defined in Design Documents
- ✓Written in Javascript
- ✓Processed by V8 JS Engine (Node.js)
- ✓*Function* receives each document and its metadata
- ✓*Emit* returns new key and new value to be indexed for this document
- ✓Get results via REST API port 8092
- ✓Numerous query parameters supported to filter results

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

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

How does N1QL ("SQL for Documents") work?

SQL for multi-dimensional, flexible data ...

```
SELECT * FROM tutorial
WHERE fname = 'Ian'
```

How does N1QL ("SQL for Documents") work?

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

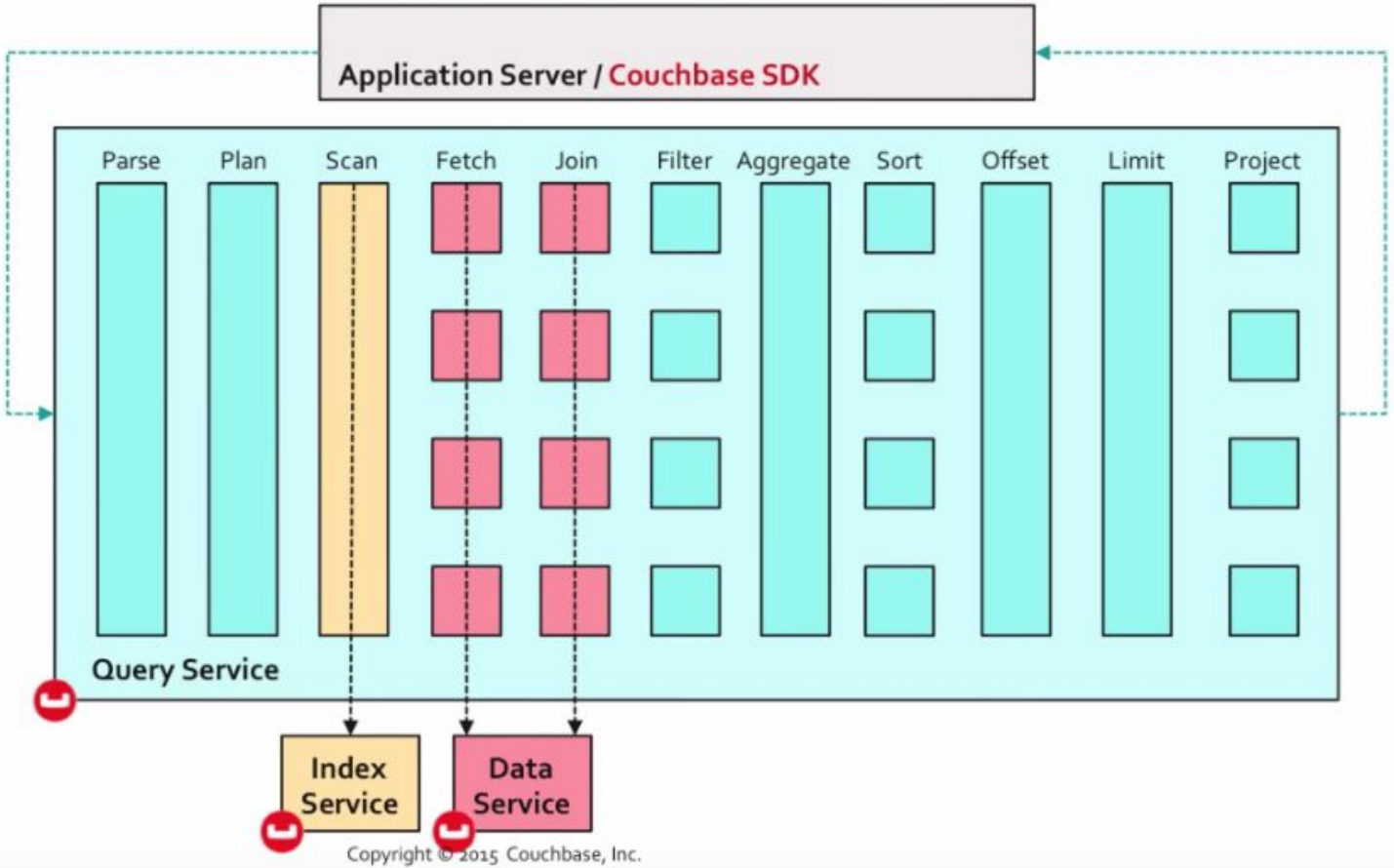
Client code simplification

Ad hoc queries

http://**query**.couchbase.com/**tutorial**

How is a query processed?

1. Request
2. Plan
   query to execution plan
3. Scan
   attributes to keys
4. Fetch
   keys to documents
5. Evaluate
   documents to results
6. Response



What we have?

Three access approaches beyond basic document read/write

- ✓ Secondary indexing – access documents by alternate keys
- ✓ Querying – access sets of documents matching conditions
- ✓ Aggregation – access summaries, counts, and other reductions

Secondary indexes can be created two ways

- ✓ Global secondary indexes provided by Index Service
- ✓ Distributed secondary indexes provided by Data Service

Query service provides N1QL, a powerful SQL superset for JSON

- ✓ N1QL queries can access either global or distributed indexes
- ✓ REST based API available to access distributed indexes ("MapReduce Views")

How is a cluster organized?

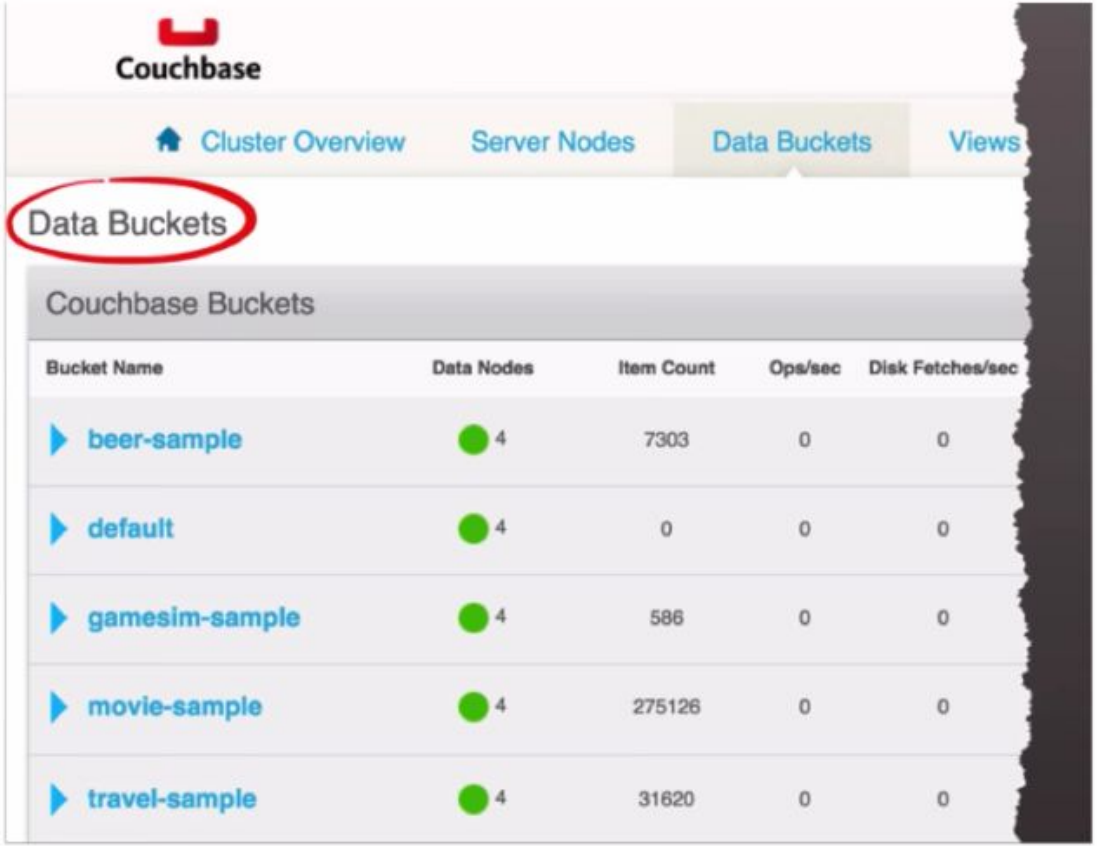
Virtual buckets tracked by the cluster map.

What is a Data Bucket?

A logical container of uniquely keyed documents

- ✓ Keyspace
- ✓ Database

So, what equates to a "table"?



What is a Data Bucket?

A logical container of uniquely keyed documents

- ✓ Keyspace
- ✓ Database

So, what equates to a "table"?

- ✓ Documents which happen to follow the same structure
- ✓ Neither needed nor enforced by Couchbase

Why more than one bucket?

- ✓ Cache management
- ✓ Replication
- ✓ Indexing

```
{
  'first_name' : 'Roberta',
  'last_name' : 'Red',
  'age' : 54,
  'admin' : true
}
{
  'first_name' : 'Betty',
  'last_name' : 'Blue',
  'age' : 36,
  'admin' : false
}
{
  'first_name' : 'Yolanda',
  'last_name' : 'Yellow',
  'logins' :
  ['2015-03-26-09:20AM', '2015-03-31-16:13PM'],
}
```

What do buckets control?

Caching and Persistence

Data is cached in RAM for immediate availability, then asynchronously persisted to disk

Each bucket shares a thread pool to handle its persistence queue

If incoming data exceeds RAM, may fully eject old documents, or retain key/metadata to speed later access

Uncached data is retrieved from disk, delivered, then cached

Replication

Each bucket shares a thread pool to handle its replication queue

To secure against machine failure, up to 3 replicas can be made of each document

Replicas never reside on the same node as their active documents

Rack awareness enables group-level control of replica placement

Rebalancing

Fast, efficient background document redistribution when nodes are added, removed, or failed over

Any replicas promoted to active status, due to failover, are re-replicated

What is a Virtual Bucket?

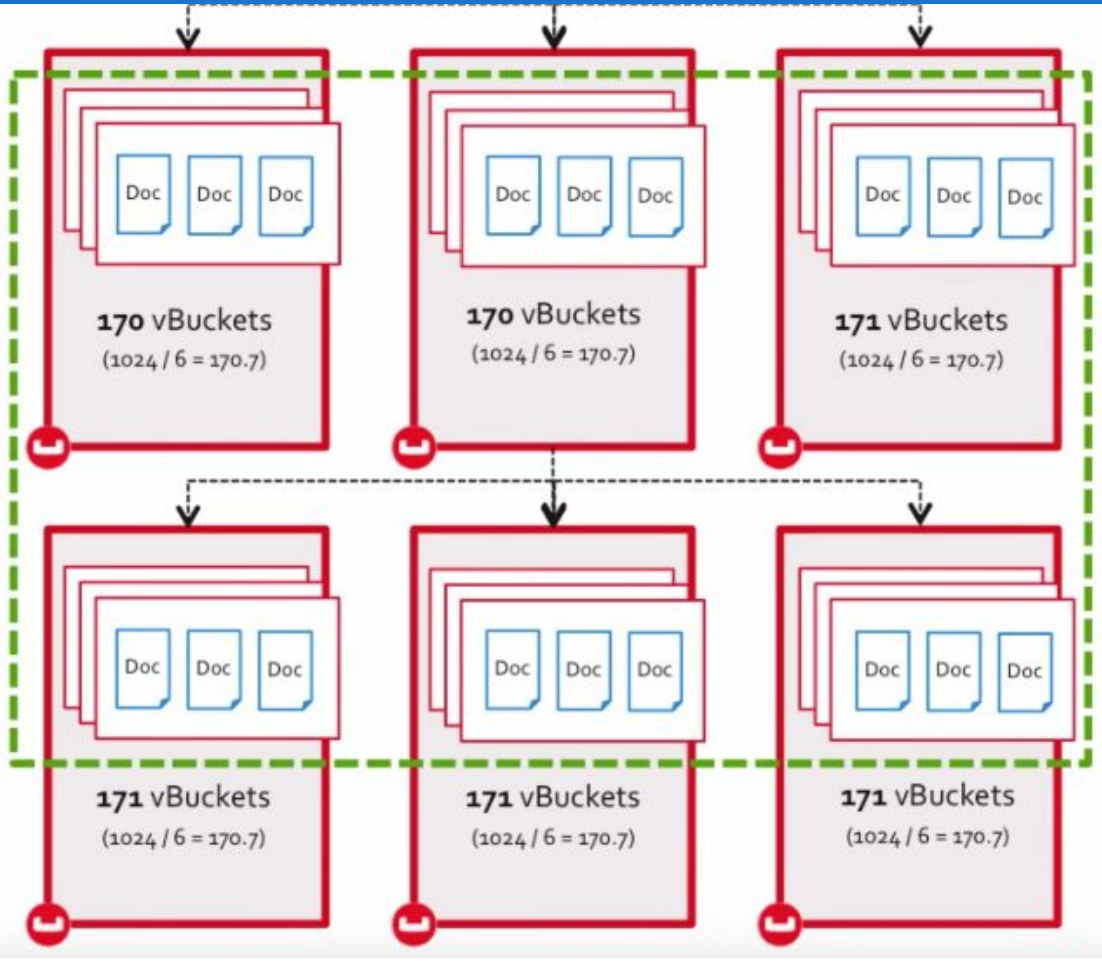
One organizational segment of a Data Bucket

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?

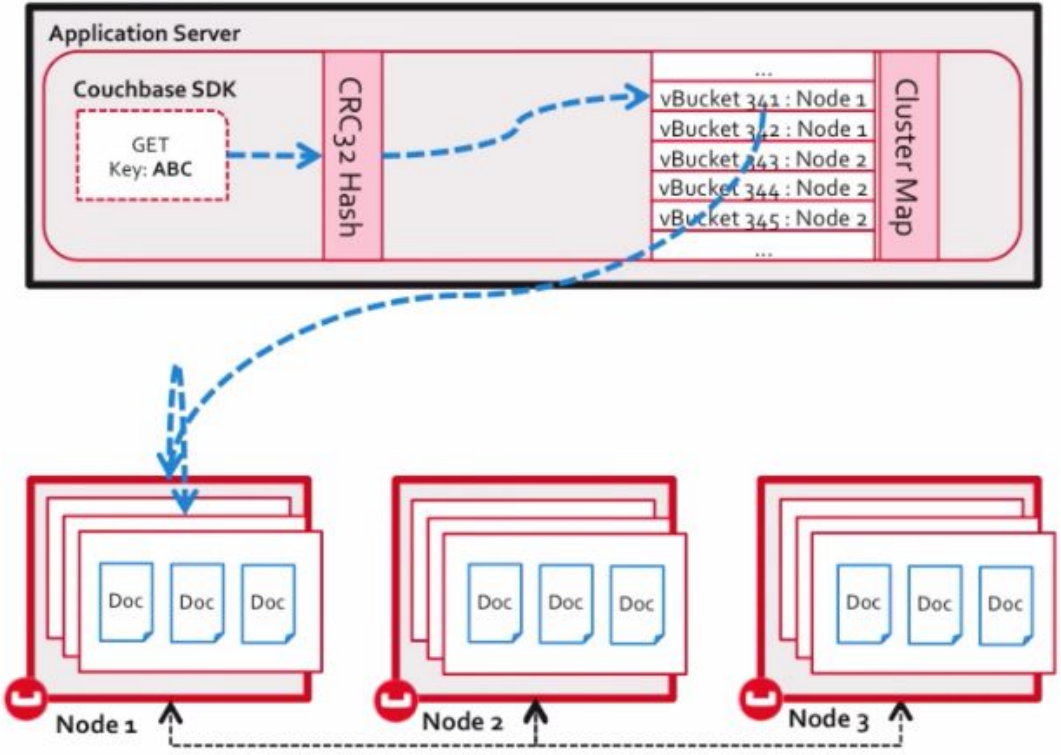
A document's location is determined by its *key*

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 client's *Cluster Map*

Cluster Map identifies the correct location for this read or write

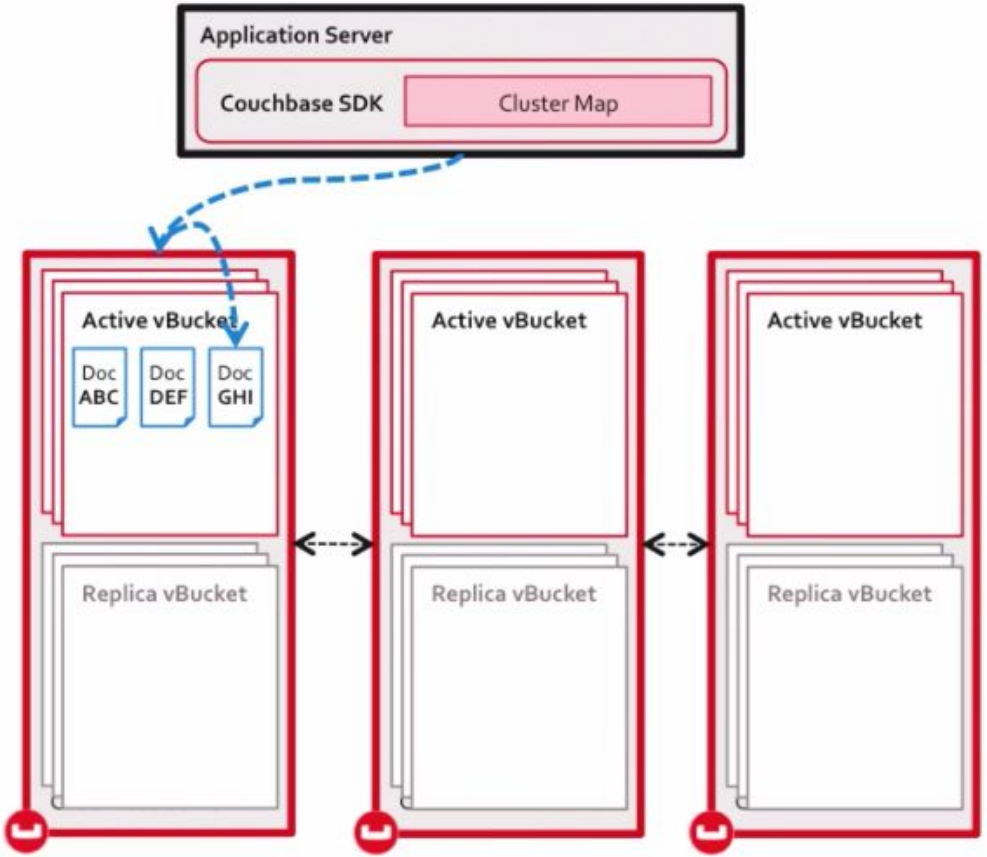
- ✓ Node
- ✓ vBucket



Hos is a write persisted and replicated?

All replication relies on DCP (Database Change Protocol)

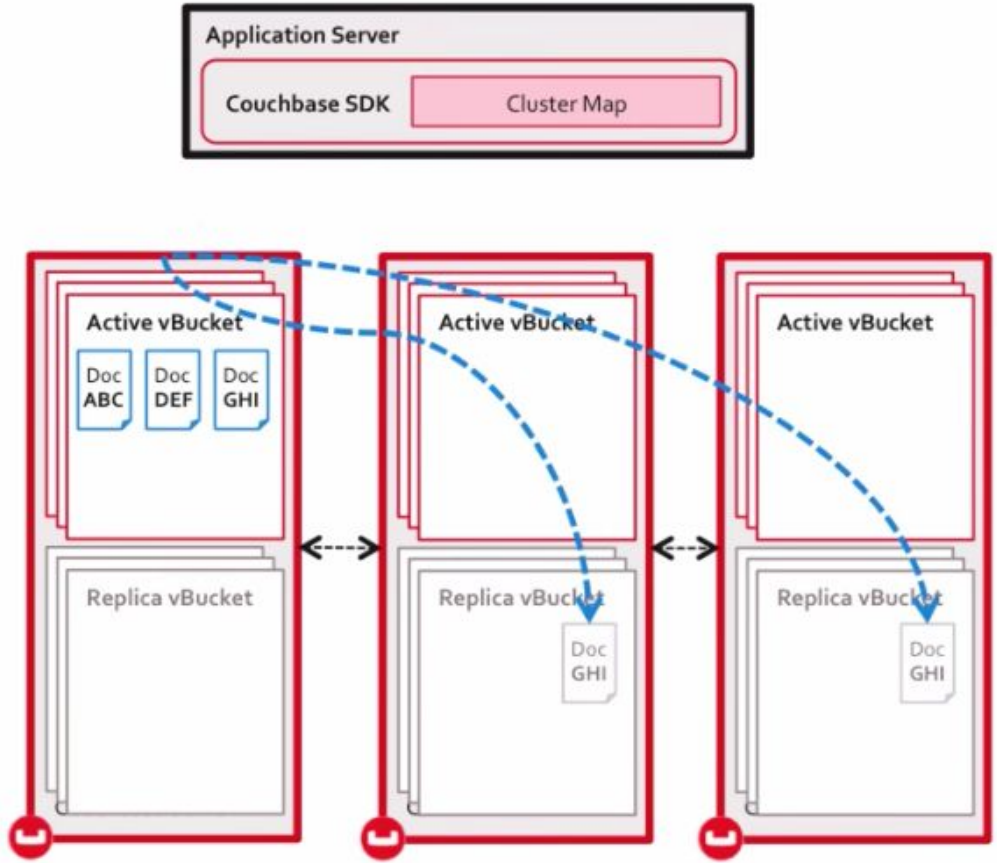
✓ Local persistence



Hos is a write persisted and replicated?

All replication relies on DCP (Database Change Protocol)

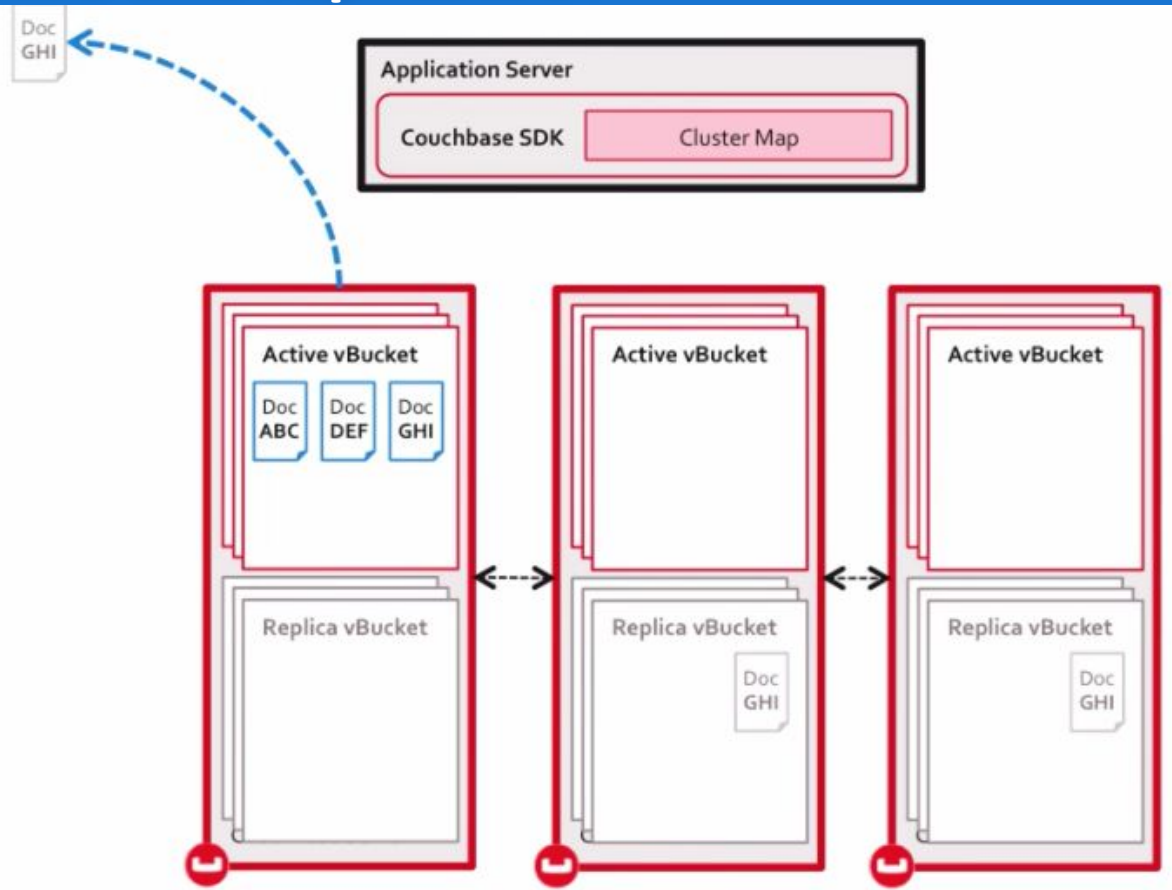
- ✓ Local persistence
- ✓ In-cluster replication



Hos is a write persisted and replicated?

All replication relies on DCP (Database Change Protocol)

- ✓ Local persistence
- ✓ In-cluster replication
- ✓ Cross datacenter replication



What we have?

A bucket is a logical keyspace of uniquely identified documents

- ✓ Divided into 1024 segments known as virtual buckets ("vBuckets")
- ✓ Distributed evenly across the nodes of the cluster

Buckets control caching, persistence, replication, and rebalancing

During read/write operations, document keys are hashed

Cluster map determines which node and vBucket hold a document

- ✓ Resides on the client as part of the running Couchbase SDK

Persistence and replication use Database Change Protocol (DCP)

- ✓ Local persistence, in-cluster replication, and XDCR
- ✓ Ordered, consistent, memory to memory, no loss recovery

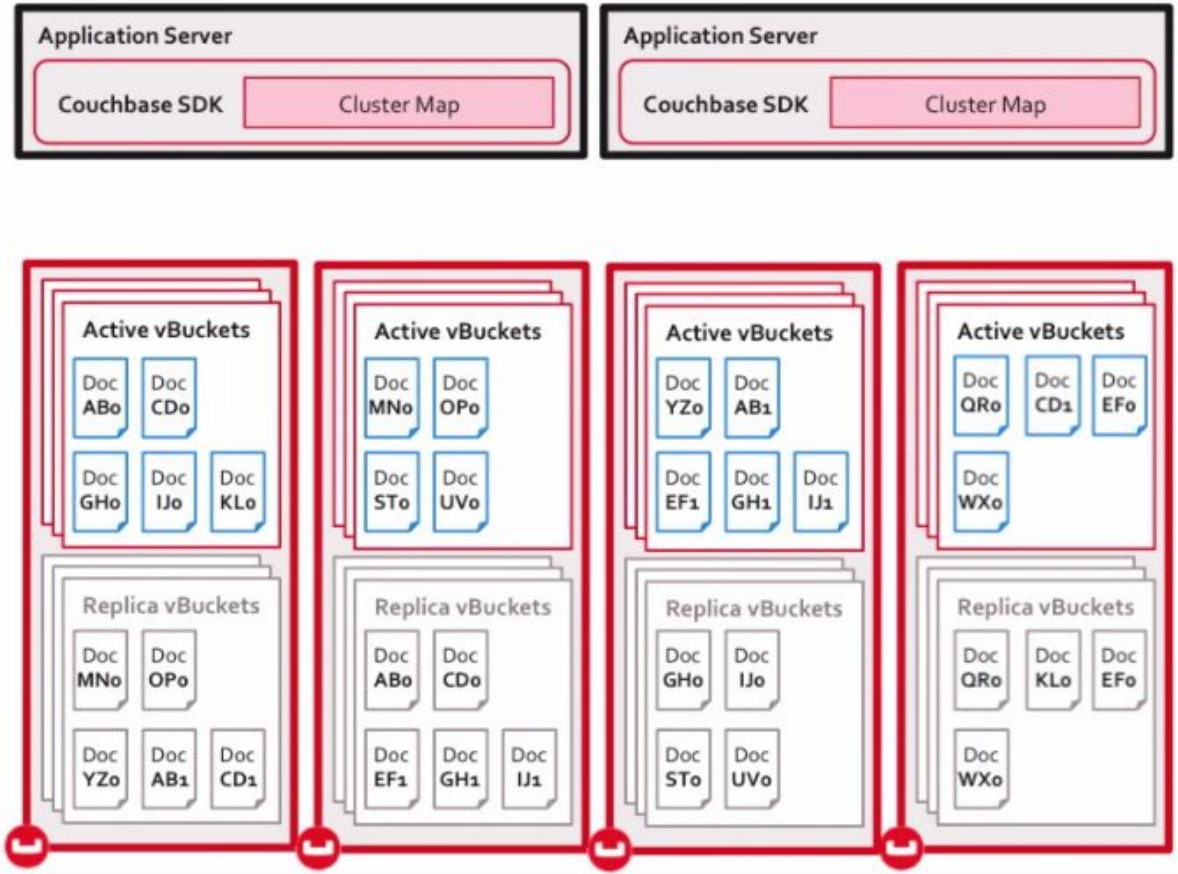
How does the cluster respond to topological changes?

Migrate the docs and update the map. Automatically.

What happens when nodes are added to a cluster?

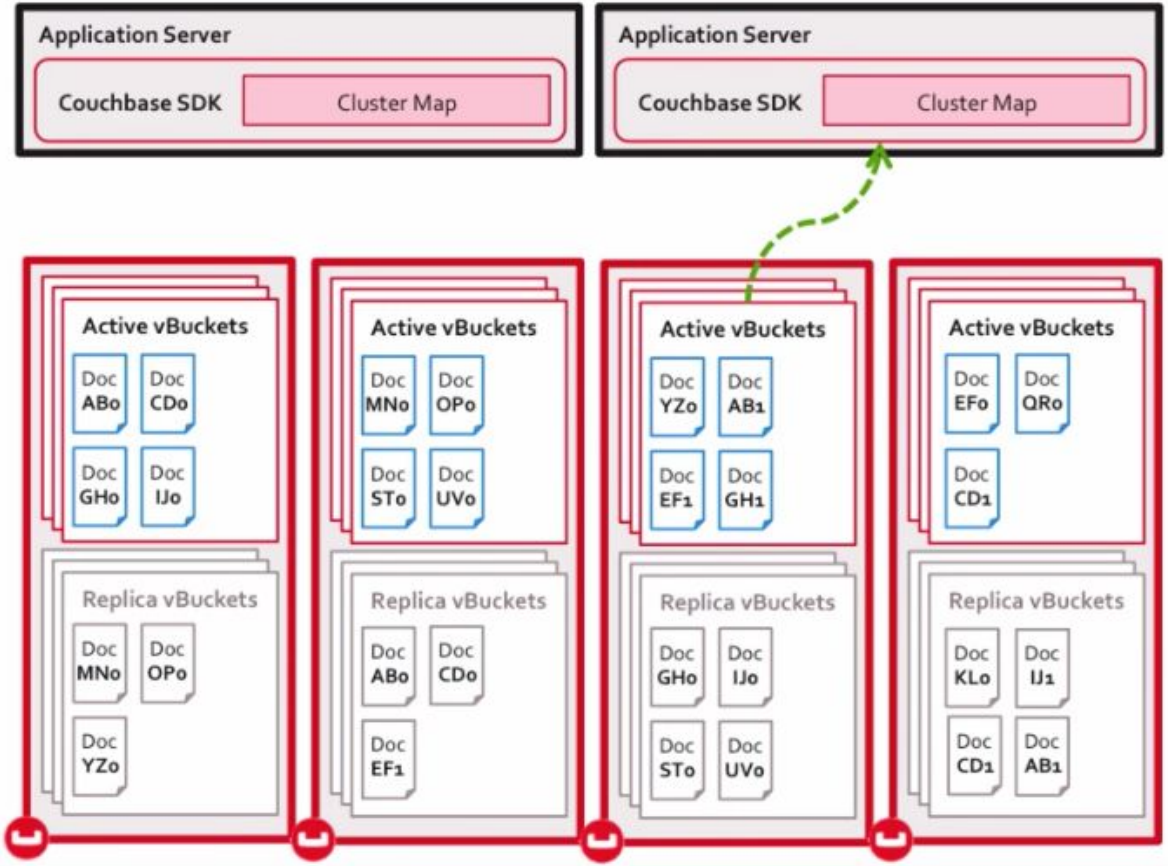
- ✓ New node address added via UI or REST
- ✓ vBuckets are recalculated for each Bucket
- ✓ Documents are incrementally transferred
- ✓ Updated cluster maps are continuously provided

Zero downtime



What happens when nodes are removed?

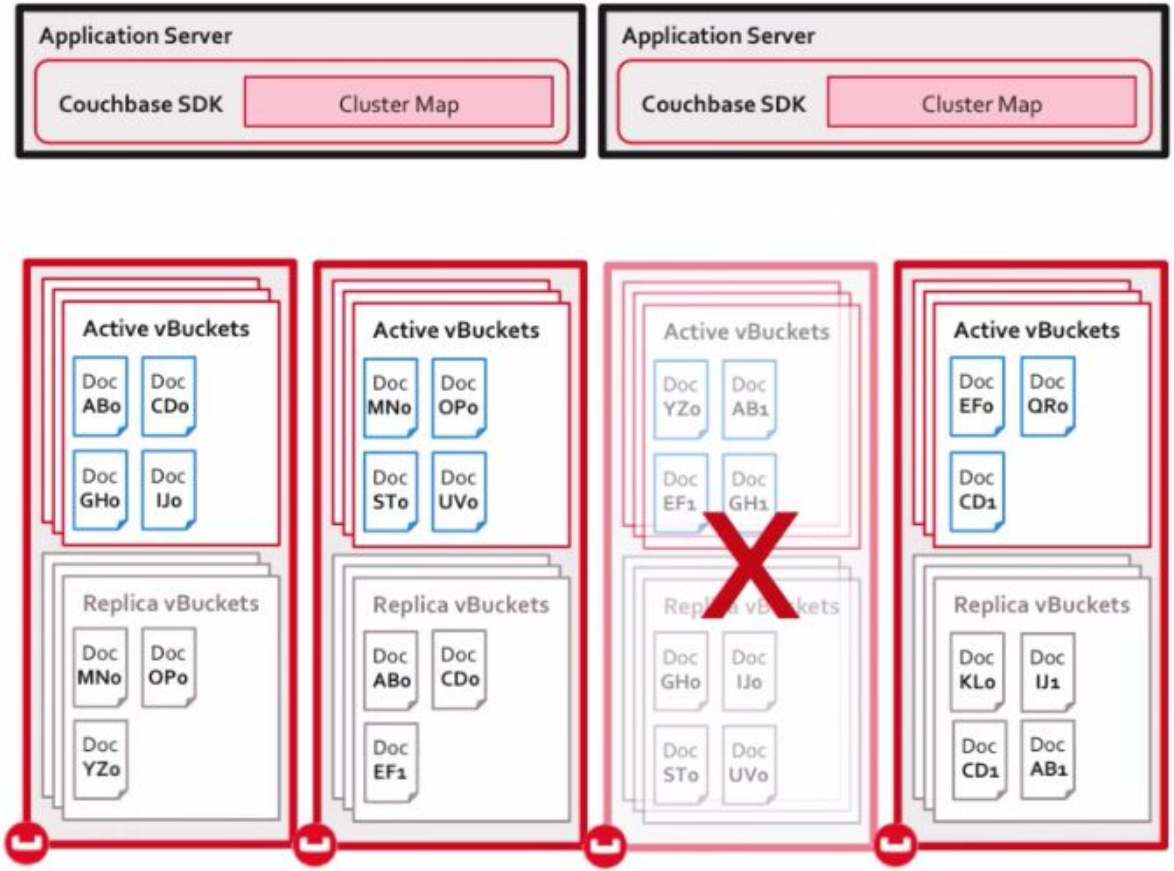
When a node fails, *failover* can be manual or automatic



What happens when nodes are removed?

When a node fails, *failover* can be manual or automatic

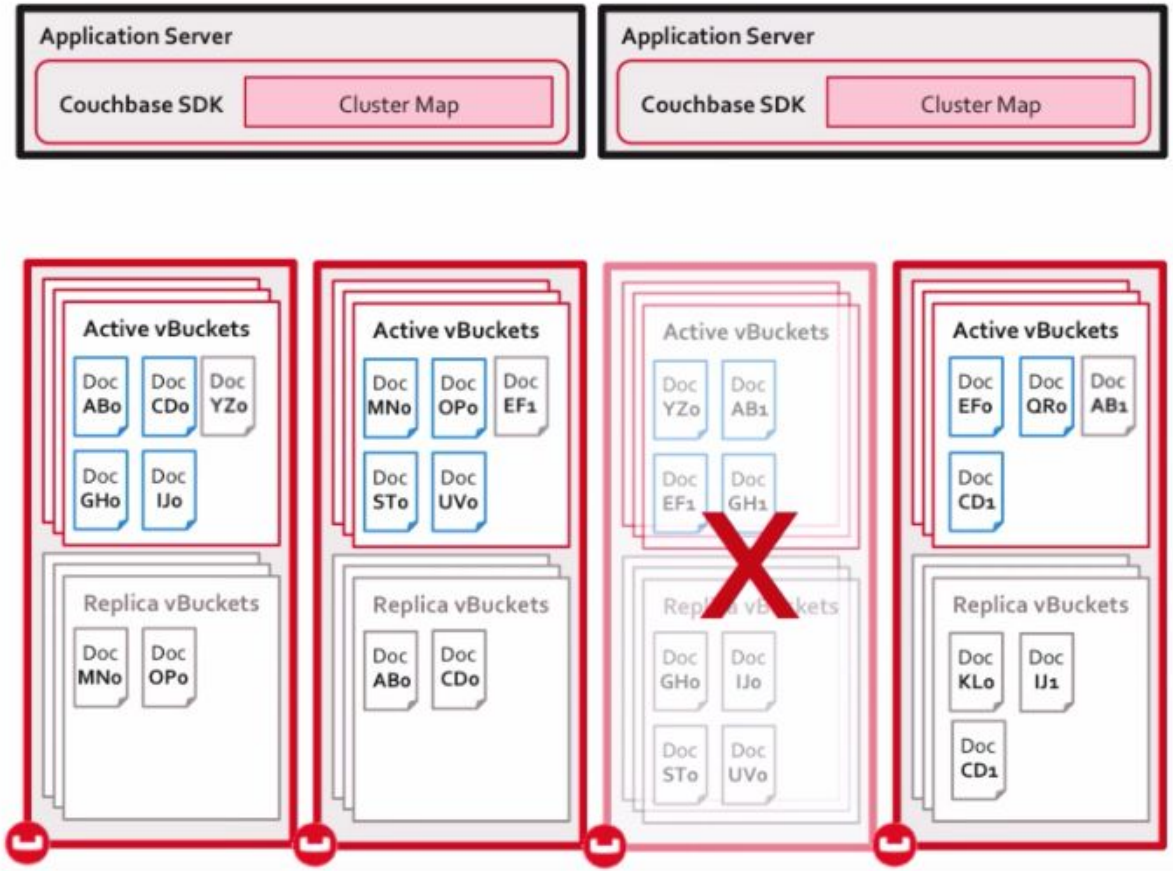
- ✓ Replicas promoted



What happens when nodes are removed?

When a node fails, *failover* can be manual or automatic

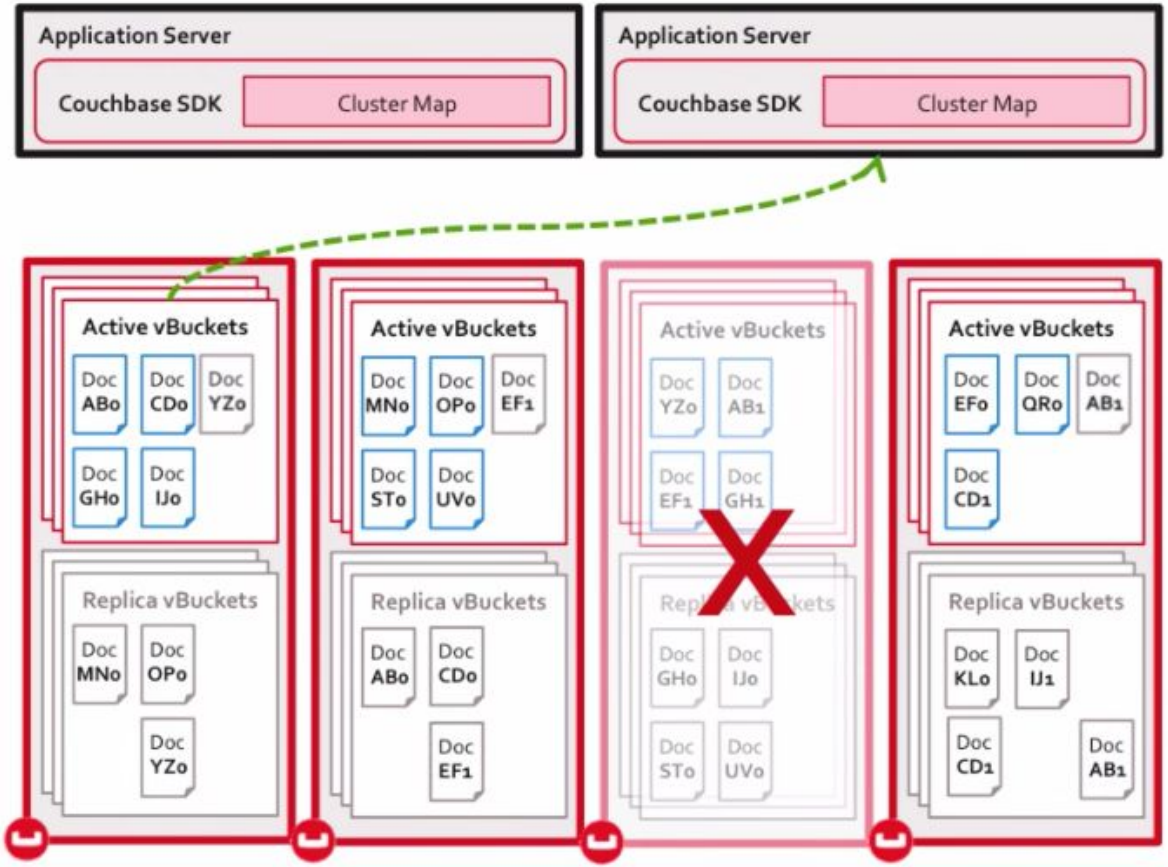
✓ Replicas promoted



What happens when nodes are removed?

When a node fails, *failover* can be manual or automatic

- ✓ Replicas promoted
- ✓ Cluster map updated
- ✓ Rebalance cluster (optional)



What we have?

Application server has single logical connection to Couchbase

- ✓ Multiple pooled connections maintained by Couchbase library
- ✓ Cluster topology abstracted by cluster map

On adding a node to the cluster

- ✓ Incremental transfer of active and replica documents
- ✓ Cluster map updated
- ✓ Zero downtime

On node failure, failover process can be manual or automatic

- ✓ Replica documents promoted but not recreated
- ✓ Cluster map updated
- ✓ Rebalance recommended

What is XDCR?
(Cross Data Center Replication)

Keep your data close, worldwide.

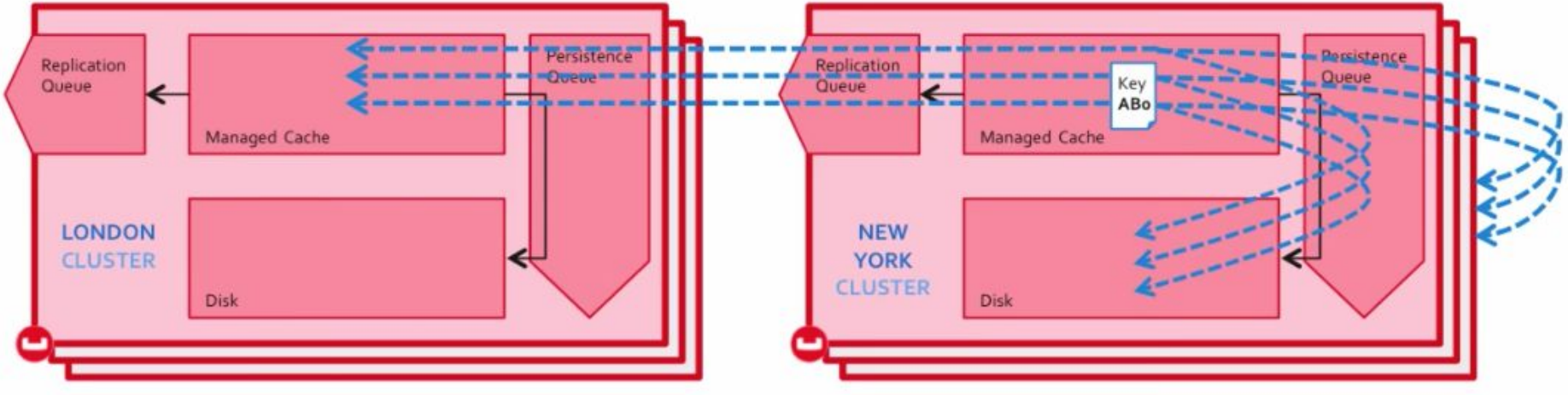
How do mutations flow through Couchbase?

All replication relies on DCP, which simultaneously provides

- ✓ Local persistence
- ✓ In-cluster replication
- ✓ Cross datacenter replication (XDCR)

DCP provides for each of these

- ✓ Ordered mutations
- ✓ Consistent data snapshots
- ✓ No-loss recovery if interrupted
- ✓ RAM to RAM streaming
- ✓ Multi-threaded, parallel processing



What is cross data center replication (XDCR)?

Secure, continuous memory-to-memory replication among clusters

Configured per bucket, mutations pushed after local persistence

32 SSL-encrypted streams (default) shuffling among vBuckets
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

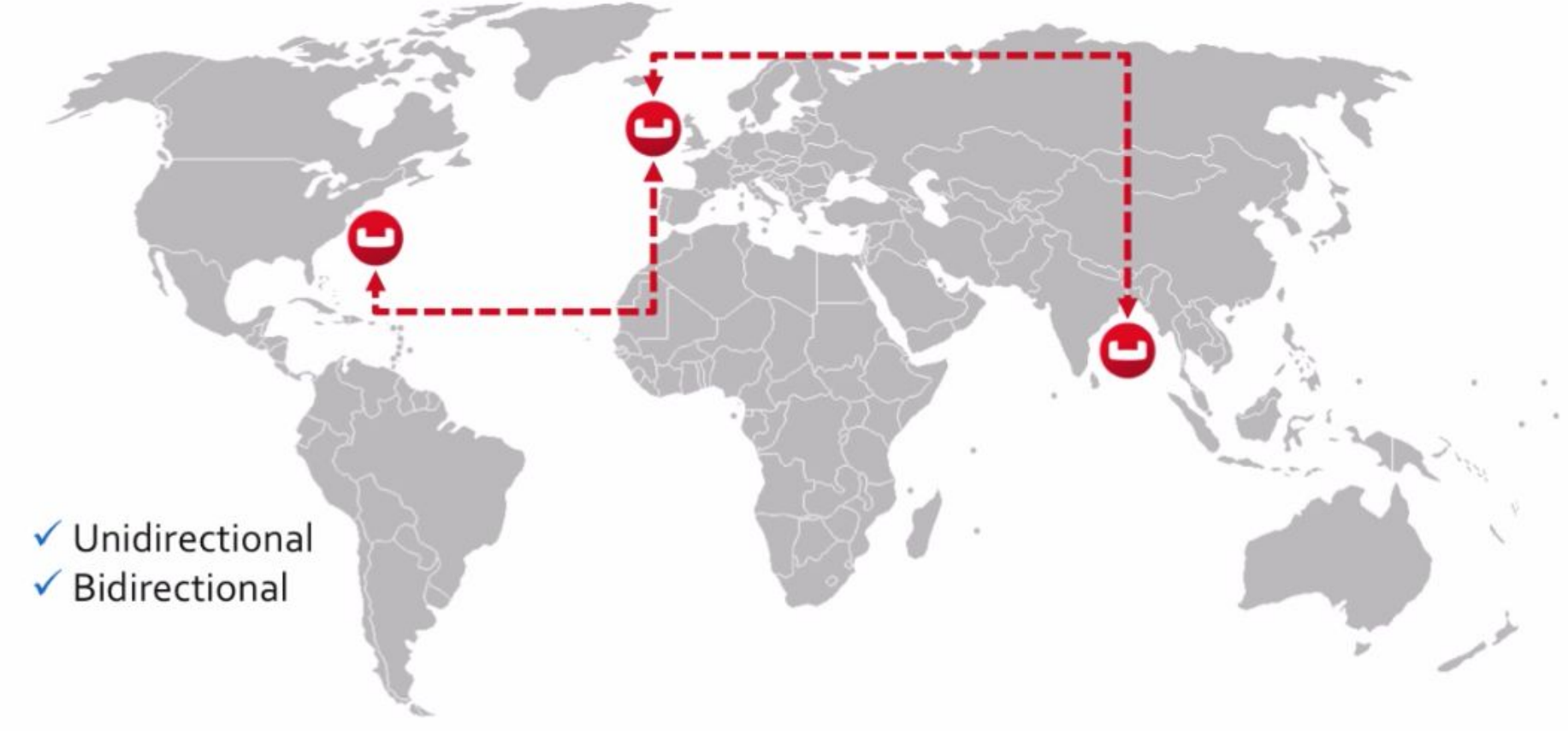
Regular checkpoints to support pause/resume

Recoveries start at most recent checkpoint

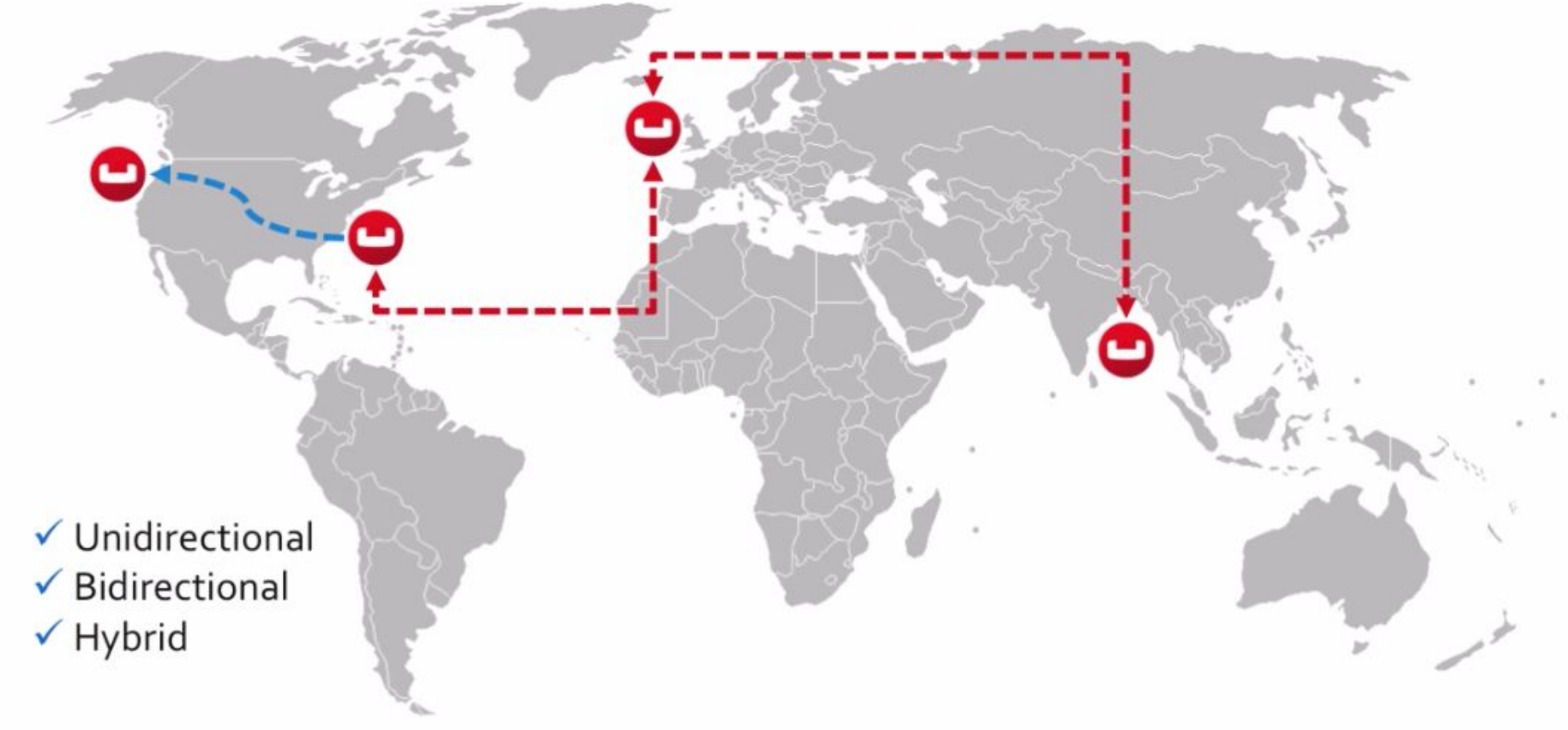
Why is XDCR used?

What topologies are available?

What topologies are available?



What topologies are available?



What we have?

All replication relies on DCP, to provide speed and consistency

Cross data center replication (XDCR) is configured per bucket

- ✓ Continuous – 32 streams (default) working among the vBuckets
- ✓ Topology aware – auto-recovery if any node fails on either end
- ✓ Efficient – if multiple writes in local queue, only most recent pushed
- ✓ Resilient – regular checkpoints, recoveries start with most recent

XDCR may be unidirectional or bidirectional, and is used for

- ✓ Regional availability
- ✓ Disaster recovery
- ✓ Pre-production test beds

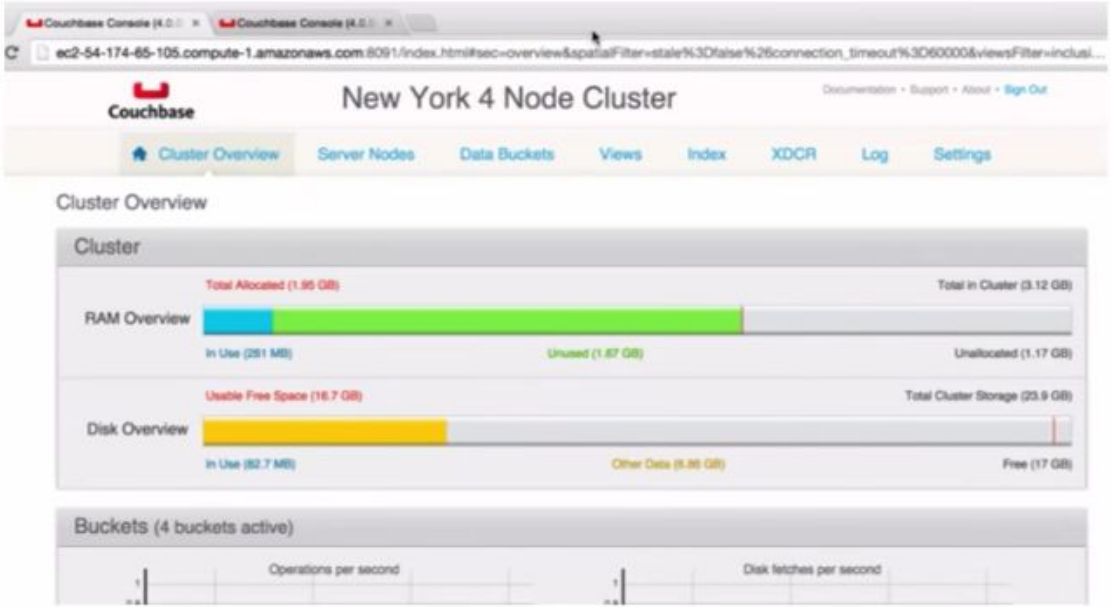
How is a cluster administered?

UI, CLI, or REST API. Take your pick.

What administrative interfaces are available?

Web interface

✓ http://<domain>:8091



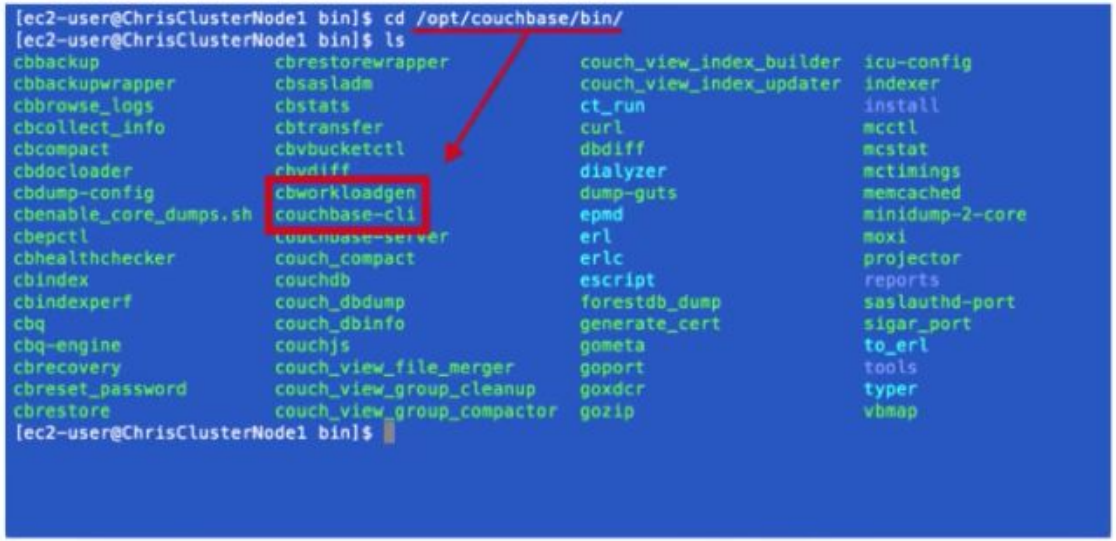
What administrative interfaces are available?

Web interface

- ✓ http://<domain>:8091

Command line tools

- ✓ couchbase-cli
- ✓ cbworkloadgen
- ✓ more ...



What administrative interfaces are available?

Web interface

✓ http://<domain>:8091

Command line tools

✓ couchbase-cli
✓ cbworkloadgen
✓ more …

REST API

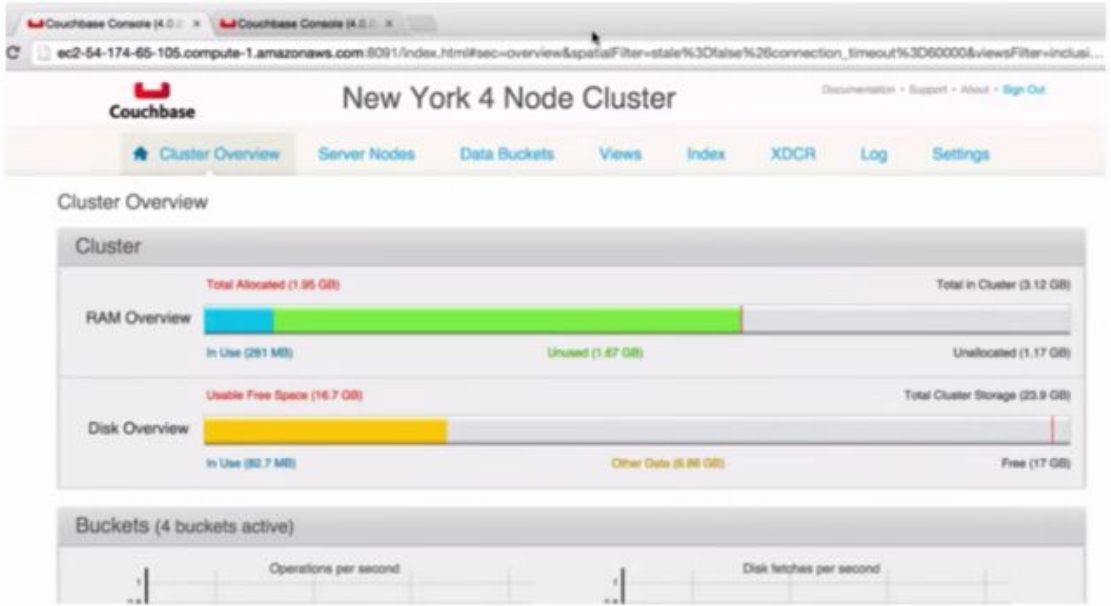
✓ entire API just a URL away …

HTTP method	URI path	Description
GET	/pools/nodes	Retrieves information about nodes in a cluster.
POST	/controller/setRecoveryType	Sets the recovery type to be performed for a node. Options are delta or full.
POST	/controller/failOver	Fails over nodes.
POST	/controller/startGracefulFailover	Sets graceful failover for a specific server node. The server node is specified with the `otpNode=[node_name]` parameter.
POST	/settings/web	Sets user names and passwords.
POST	/pools/default memoryQuota	The `memoryQuota` parameter sets the memory quota.
POST	/nodes/self/controller/settings	Sets the path for index files.
GET	/pools/default/buckets/default/nodes/[host]:[port]/stats	Retrieves statistics for a node.

How do you navigate the web interface?

Couchbase Console Tour

- ✓ Navigate the interface
- ✓ Create a new data bucket
- ✓ Create a cluster reference
- ✓ Create and run XDCR job
- ✓ Survey available metrics



What is the primary command line tool?

```
[ec2-user@ChrisClusterNode1 bin]$
[ec2-user@ChrisClusterNode1 bin]$ ./couchbase-cli

CLUSTER:
  --cluster=HOST[:PORT] or -c HOST[:PORT]

OPTIONS:
  -u USERNAME, --user=USERNAME      admin username of the cluster
  -p PASSWORD, --password=PASSWORD  admin password of the cluster
  -o KIND, --output=KIND            KIND is json or standard
  -d, --debug
  -s, --ssl                         uses SSL for communication with secure servers

usage: couchbase-cli COMMAND CLUSTER [OPTIONS]

COMMANDs include
  bucket-compact        compact database and index data
  bucket-create         add a new bucket to the cluster
  bucket-delete         delete an existing bucket

  ssl-manage            manage cluster certificate
  user-manage           manage read only user
  xdcr-replicate        xdcr operations
  xdcr-setup            set up XDCR connection
```

How do you navigate the command line tool?

couchbase-cli tour

- ✓ Navigate to tools
- ✓ View bucket list
  bucket-list
- ✓ View servers in cluster
  server-list

```
[ec2-user@ChrisClusterNode1 bin]$
[ec2-user@ChrisClusterNode1 bin]$ ./couchbase-cli

CLUSTER:
  --cluster=HOST[:PORT] or -c HOST[:PORT]

OPTIONS:
  -u USERNAME, --user=USERNAME      admin username of the cluster
  -p PASSWORD, --password=PASSWORD  admin password of the cluster
  -o KIND, --output=KIND            KIND is json or standard
  -d, --debug
  -s, --ssl                         uses SSL for communication with secure servers

usage: couchbase-cli COMMAND CLUSTER [OPTIONS]

COMMANDs include
  bucket-compact        compact database and index data
  bucket-create         add a new bucket to the cluster
  bucket-delete         delete an existing bucket

  ssl-manage            manage cluster certificate
  user-manage           manage read only user
  xdcr-replicate        xdcr operations
  xdcr-setup            set up XDCR connection
```

What further tools are available?

- ✓ **cbq** – execute N1QL ("SQL for JSON") queries from the command line
- ✓ **cbanalyze-core** – parse and analyze core dump data
- ✓ **cbbackup** – copy data from an entire cluster/bucket, or a single node/bucket
- ✓ **cbcollect_info** – provides detailed statistics for a specific node
- ✓ **cbdocloader** – load JSON documents from a directory or single .zip file
- ✓ **cbreset_password** – reset an administrative or user read-only password
- ✓ **cbrestore** – restores data from a file to an entire cluster or a single bucket
- ✓ **cbstats** – get node and cluster-level stats about performance and storage
- ✓ **cbworkloadgen** – generate random data and perform read/writes
- ✓ … many more community tools

Where do you find these on a node?

- ✓ **Linux**
  /opt/couchbase/bin
  /opt/couchbase/bin/install
  /opt/couchbase/bin/tools

- ✓ **Windows**
  C:\Program Files\couchbase\server\bin
  C:\Program Files\couchbase\server\bin\install
  C:\Program Files\couchbase\server\bin\tools

- ✓ **Mac OSX**
  /Applications/Couchbase Server.app/Contents/Resources/couchbase-core/bin

What we have?

Couchbase provides three approaches to administration

- ✓ Web interface
- ✓ Command line tools
- ✓ REST API

Couchbase provides comprehensive command line tooling

The web interface and command line tools rely on the REST API

REST API enables easy integration into 3rd party admin tools

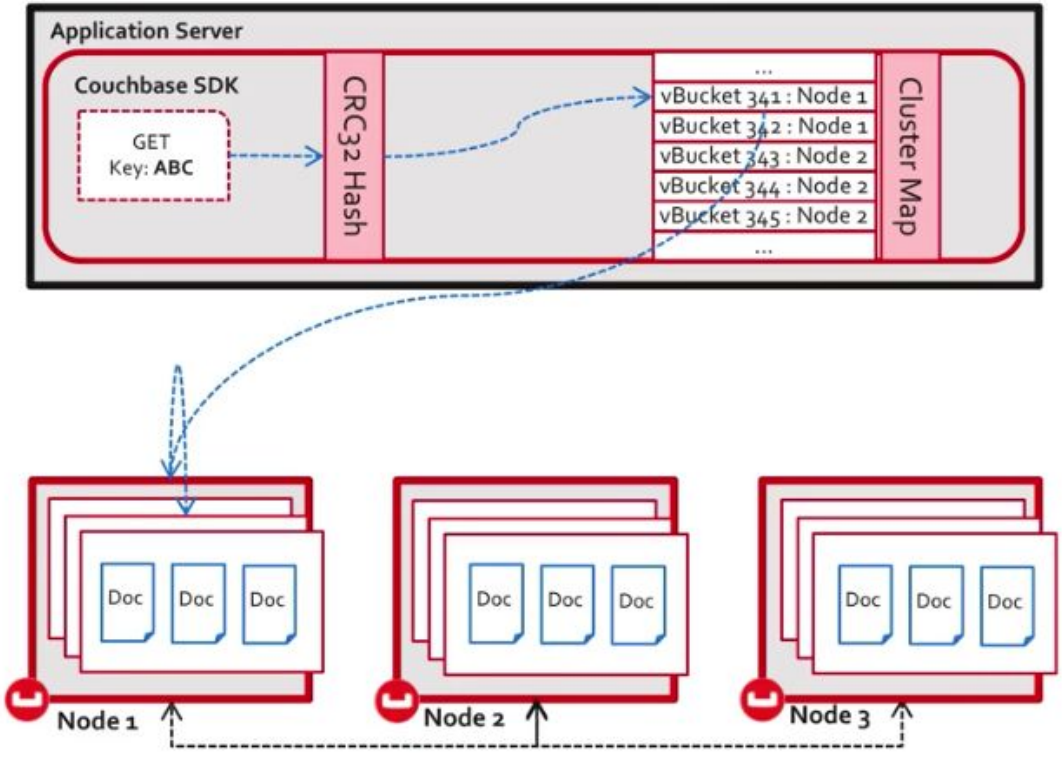
How does Couchbase support developers?

Open source SDKs for major languages.

what is the role of the Couchbase SDK?

SDKs provide core capabilities for NoSQL applications

- ✓ Pooled cluster connections
- ✓ Encryption on-the-wire
- ✓ JSON (de)serialization
- ✓ Document create, read, update, and delete via N1QL or API call
- ✓ Document index and query via N1QL or MapReduce Views
- ✓ Choice of synchronous or asynchronous APIs
- ✓ Single point of contact regardless of changes to cluster topology



Fore what languages are SDKs supported?

Java

.NET

Node.js

PHP

C

Python

Go

http://www.couchbase.com/developers

What are the common APIs?

The following are implemented across all Couchbase SDKs

- ✓ **insert** – add key/document or key/binary pair to bucket, fail if exists
- ✓ **upsert** – add key/document or key/binary pair to bucket, success if exists
- ✓ **replace** – replace document/binary at specified key, fail if it does not exist
- ✓ **remove** – delete (tombstone) value at specified key, fail if it does not exist
- ✓ **query** – execute N1QL: *select, insert, update, delete, create index, where, join, order by,* etc …
- ✓ **append** – append to current document at key
- ✓ **prepend** – prepend to current document at key
- ✓ **get** – retrieve document/binary at key
- ✓ **getAndLock** – get document/binary at key, then lock (change CAS)
- ✓ **unlock** – unlock previously locked document/binary at key
- ✓ **touch** – update time to live (TTL) for value at key
- ✓ **getAndTouch** – get document/binary at key, and update its time to live
- ✓ **counter** – increment or decrement a key's numeric value

Show me some code ...

Connect, create, and retrieve a document (Java)

- ✓ Create connection
- ✓ Open bucket
- ✓ Create JSON object
- ✓ Create doc from object, assign key, and "upsert"
  - if exists, **up**date
  - if not, in**sert**
- ✓ Get the document by key, and display

```
// Connect to localhost
Cluster cluster = CouchbaseCluster.create();

// Open a bucket connection
Bucket bucket = cluster.openBucket("customers");

// Create a document
JsonObject user = JsonObject.empty()
  .put("first", "Walter")
  .put("last", "White")
  .put("job", "chemistry teacher")
  .put("age", 50);

// Store a document
JsonDocument stored =
    bucket.upsert(JsonDocument.create("walter", user));

// Get the document
JsonDocument walter = bucket.get("walter");
System.out.println("Found: " + walter.getString("job"));
```

What about best practices?

Couchbase supports Reactive Programming

Best practices in scalably handling large data streams

Responsive. Resilient. Elastic. Message Driven.
http://www.**reactivemanifesto**.org

How about cross-project code reuse?

N1QL

"Nickel" – a structured query language for JSON

```
SELECT WHERE JOIN HAVING
CREATE INDEX
MIN MAX COUNT
INSERT UPDATE DELETE
much more ...
```

Couchbase SDKs provide language-specific DSL
to aid query construction

How about community support?

Couchbase actively and enthusiastically participates in the open source movement

Couchbase Server

Couchbase SDKs

Couchbase Mobile

http://www.couchbase.com/**open-source**

What we have?

SDK manages connections, topology, documents, and queries

Couchbase SDKs are available for

- ✓ Java
- ✓ .NET
- ✓ Node.js
- ✓ PHP
- ✓ C
- ✓ Python
- ✓ Go

N1QL is SQL for JSON

Numerous open source projects are supported by Couchbase

How does Couchbase Enterprise Edition secure your data?

Keep it secret. Keep it safe.

What security aspects are covered by Couchbase?

Authentication

Who are you?

Authorization

What are you allowed to do?

Encryption

How is the data secured?

Auditing

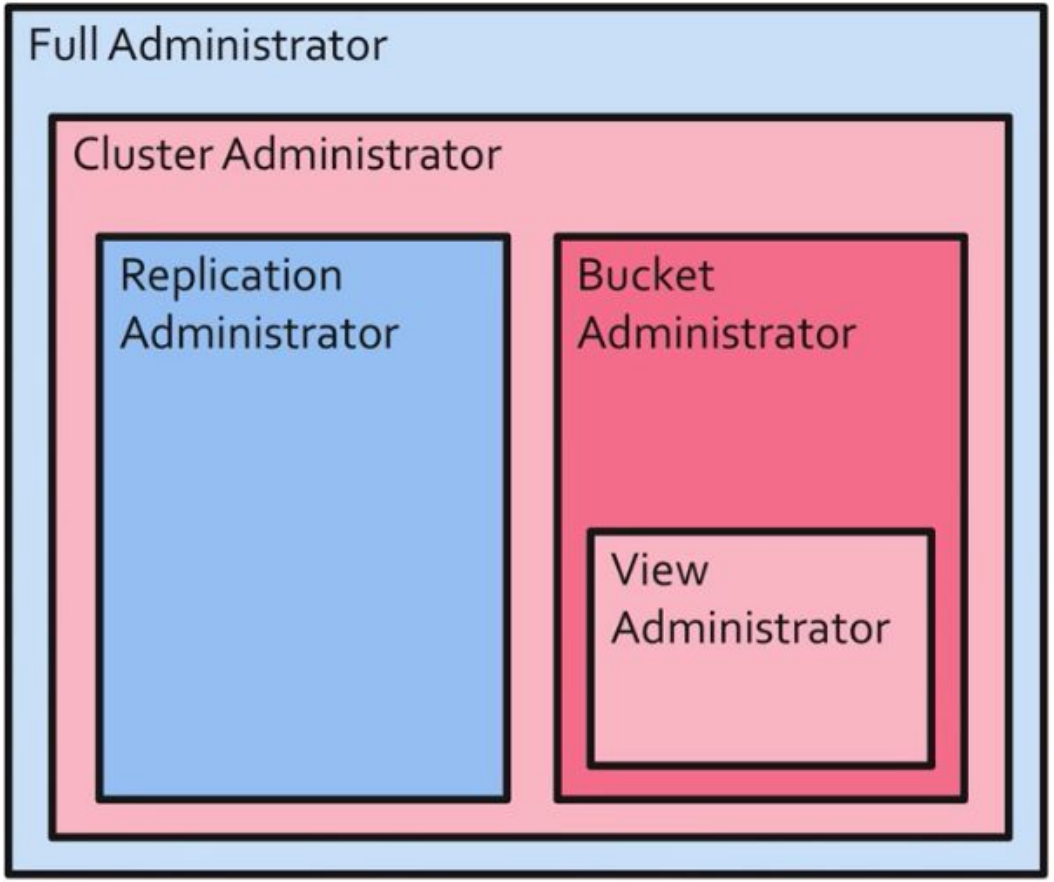
Who did what and when?

How is access controlled by user role?

Authentication via existing LDAP

Authorization via six roles

- ✓ **Full** – control all settings, including security roles
  - ✓ **Read Only** – view all settings
- ✓ **Cluster** – control all cluster settings, except security roles
- ✓ **Bucket** – control all bucket settings, except XDCR creation
- ✓ **View** – define/run map-reduce and spatial views on a bucket
- ✓ **Replication** – configure XDCR topology and replications with no other admin access



How is administrative data secured in motion?

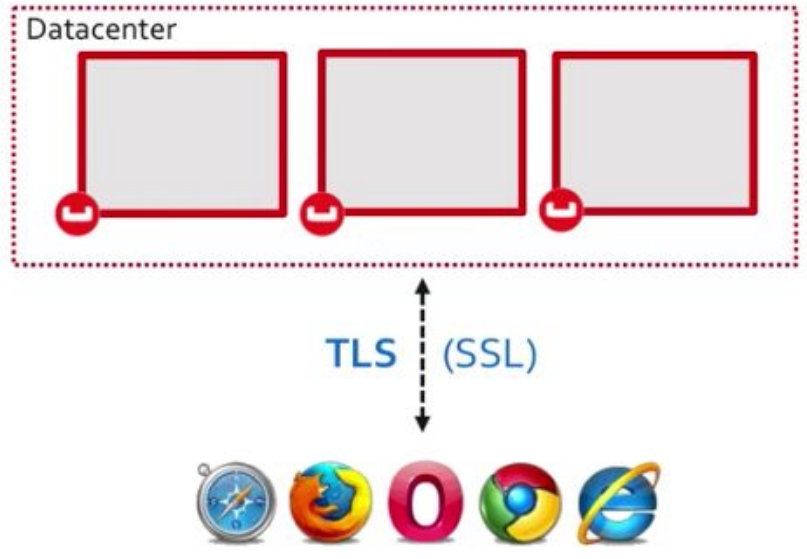
Administrative access

Encryption via TLS (SSL)

- ✓ Web UI
- ✓ REST API

Passwords are configurable for

- ✓ Administration
- ✓ Each Bucket



Couchbase Enterprise Edition supports

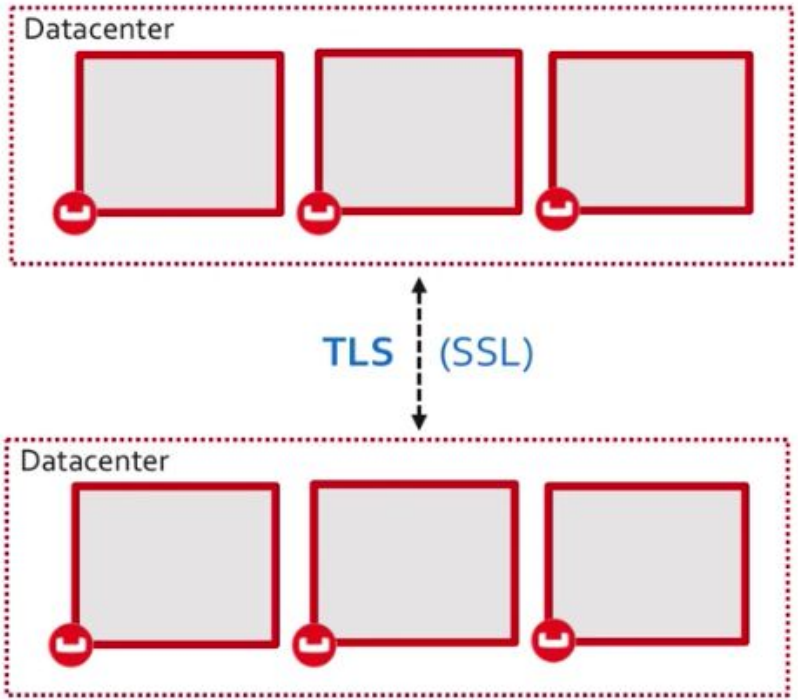
Trusted CA X.509 certificates

Self-signed certificates available in Community Edition

How is client data secured in motion?

XDCR (Server to Server) access

Encryption via TLS (SSL)



Couchbase Enterprise Edition supports

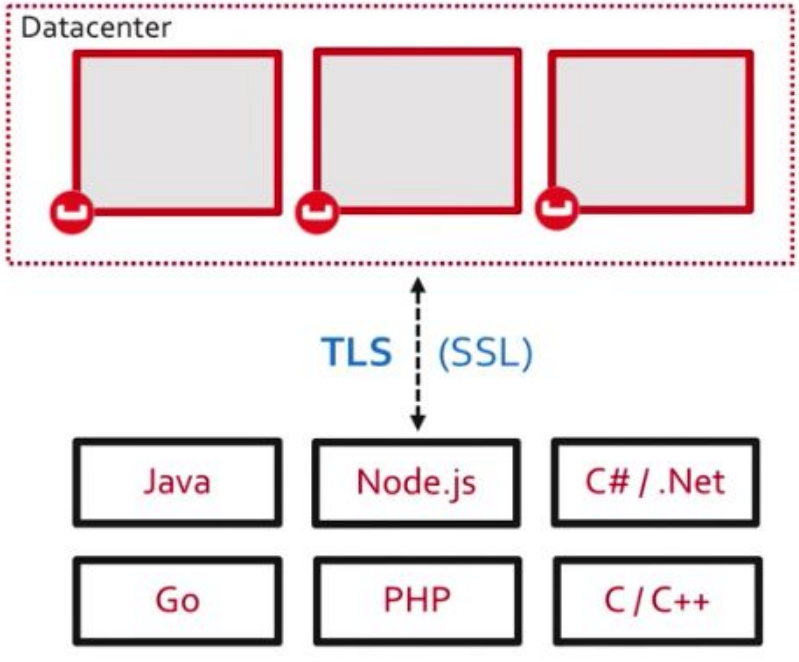
Trusted CA X.509 certificates

Self-signed certificates available in Community Edition

How is client data secured in motion?

SDK (Client to Server) access

Encryption via TLS (SSL)



Couchbase Enterprise Edition supports

Trusted CA X.509 certificates

Self-signed certificates available in Community Edition

How is client data secured at rest?

On disk storage

Encryption of all critical storage

- ✓ Data
- ✓ Indexes
- ✓ Tools
- ✓ Password files
- ✓ Configuration
- ✓ Logs

Couchbase Enterprise Edition

Fully transparent integration

Hos is administrative activity audited?

Auditing of all admin activity

- ✓ 25 named event types are tracked
- ✓ Events can only be created by Couchbase Server process
- ✓ Stored in easily filtered JSON
- ✓ Reportable using Splunk and similar tools


```
{
  "timestamp":"2015-02-20T08:48:49.408-08:00",
  "id":8192,
  "name":"login success",
  "description":"Successful login to couchbase cluster",
  "role":"admin",
  "real_userid":
  {
    "source":"ns_server",
    "user":"bjones"
  },
  "sessionid":"0fd0b5305d1561ca2b10f9d795819b2e",
  "remote":
  {
    "ip":"172.23.107.165",
    "port":59383
  }
}
```

What we have?

Authentication via existing LDAP

Authorization via six roles

- ✓ Full, Read Only, Cluster, Bucket, Replication, View

Encryption via TLS (SSL) of

- ✓ All admin traffic (Web UI, and REST API)
- ✓ All XDCR (Server to Server) traffic
- ✓ All SDK (Client to Server) traffic

Couchbase Enterprise Edition

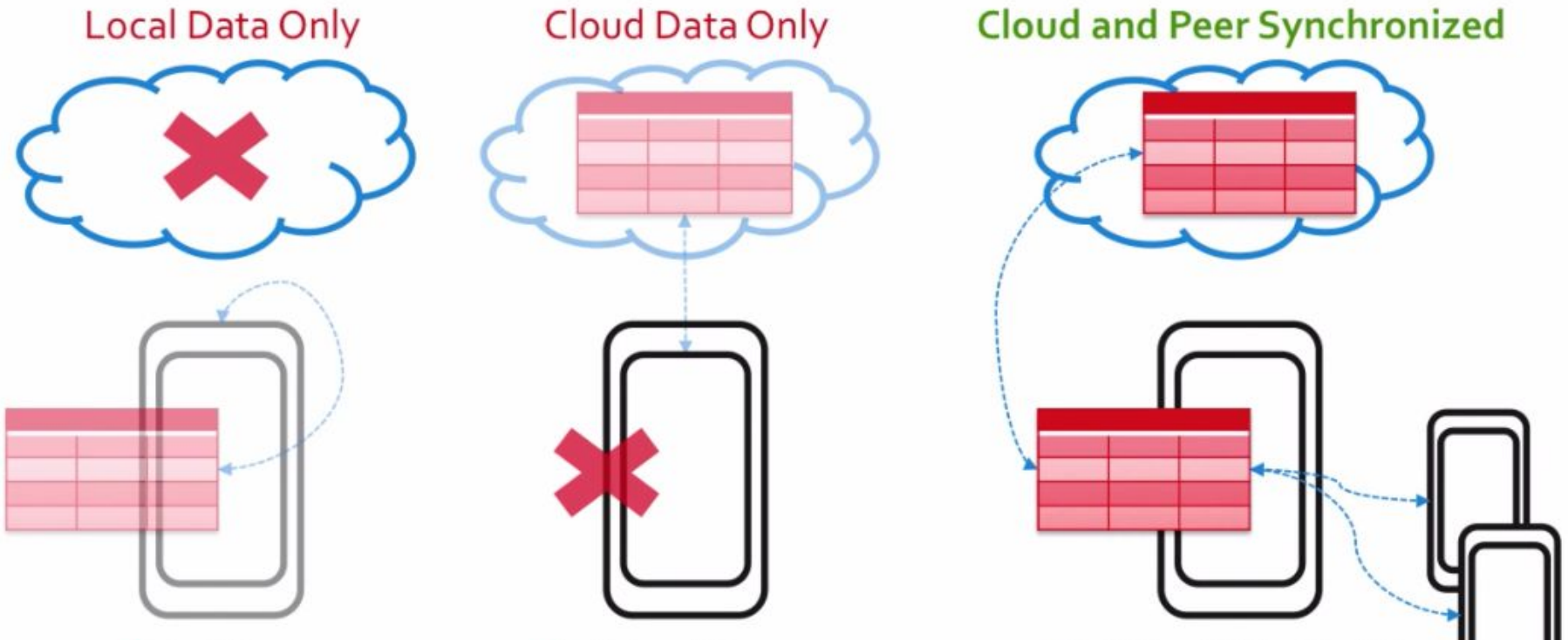
- ✓ Supports CA X.509 certificates
- ✓ Integrates to encrypt data, indexes, tools, passwords, config and log files

All administrative events are securely audited

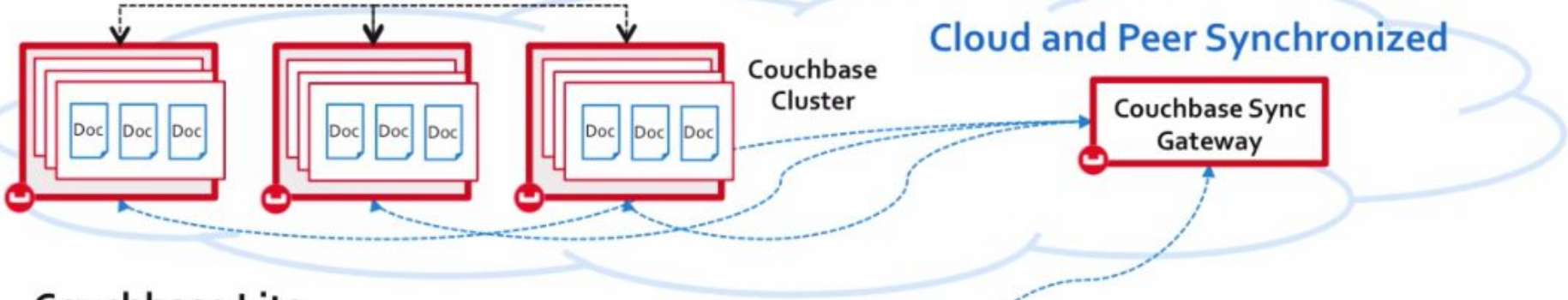
How does Couchbase optimized for mobile development?

Cloud and local synchronized, for an alway-on experience.

Why is mobile data synchronization critical?



How does Couchbase solve this problem?

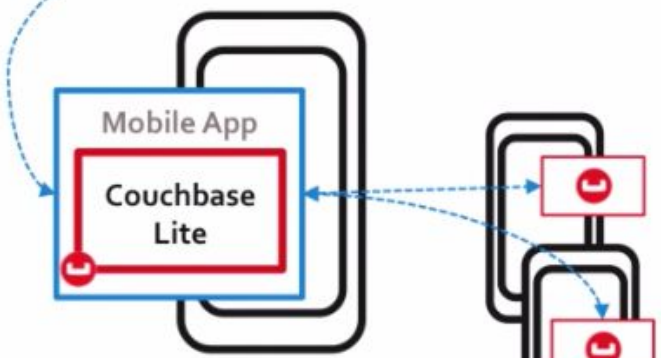


Couchbase Lite

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

Couchbase Sync Gateway

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



- ✓ Online/Offline Usability
- ✓ Expected Behavior

What all can I do with Couchbase Lite?

Create, read, update, and delete documents

Store attachments

Listen for database changes

Provide equally fast data access, online or off

Replicate directly from peer-to-peer

Where can Couchbase Lite be used?

100% open source under the Apache 2 License

http://mobile.couchbase.com

http://github.com/couchbase

What we have?

Local-only storage leads to *data* loss

Cloud-only storage leads to *user* loss

The mobile market expects both, synchronized

Couchbase Lite and Couchbase Sync Gateway solve this problem

- ✓ Full CRUD operations
- ✓ Store attachments
- ✓ React to remote changes
- ✓ Equally fast access, online or off
- ✓ Persist to server,or replicate peer-to-peer
- ✓ 100% open source, for virtually all mobile platforms

How does Couchbase optimized for mobile development?

Cloud and local synchronized, for an alway-on experience.

Big Data Analytics: Apache Hadoop

Couchbase Hadoop connector

Leverages Apache Sqoop to stream operational data to HDFS

Supported for

- ✓ Cloudera
- ✓ Hortonworks
- ✓ MapR

cloudera

Hortonworks®

MAPR

Couchbase Apache Spark connector

Streams operational data for analysis

Both Couchbase and Apache Spark can be fully memory resident

Big Data Streaming: Apache Kafka

Couchbase Apache Kafka connector

Pipe operational data streams
from Couchbase through Apache Kafka

Full Text Search: Apache Solr and beyond

Couchbase Apache Solr connector

Couchbase Plug-in for Elasticsearch

Topology aware
Continuous real-time replication

CBFT: Couchbase Full Text Search

Developer Preview in Couchbase 4.5

Cloud: Infrastructural Services

Amazon Web Services Marketplace

Microsoft Azure Marketplace

Google Cloud Platform Products & Services

Cloud: Platform as a Service (PAAS)

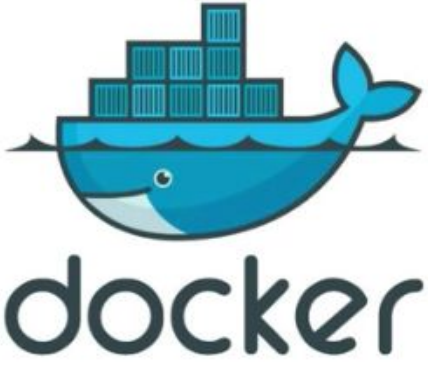
Joyent

Container managed Couchbase clusters delivered as a service

Red Hat Openshift

Couchbase deployment via Docker/Kubernetes

Cloud: Containers



https://hub.docker.com/r/couchbase/server/

Business Intelligence: ODBC/JDBC drivers

Couchbase ODBC/JDBC drivers

Developed by Simba Technologies

Supports business analysts using

- ✓ Informatica
- ✓ Tableau
- ✓ Excel
- ✓ QlikView
- ✓ SAP Lumira
- ✓ Looker
- ✓ … Huge ODBC/JDBC ecosystem

Enterprise Deployment: Monitoring

New Relic Plugin for Couchbase

Expose cluster metrics for monitoring

New Relic®

Couchbase Monitoring Extension for AppDynamics

Retrieve cluster metrics for monitoring

APPDYNAMICS

Spring Data for Couchbase

Provides familiar and consistent Spring-based programming model for Java developers

Enterprise Development: ODM for Node.js

Ottoman Object-Document Mapper for Node.js and Couchbase

Provides the foundation for efficiently building Node.js applications with Couchbase

Enterprise Development: Linq2Couchabse for .NET

Linq2Couchbase for .NET developers

Provides familiar and consistent LINQ-based programming model for .NET developers

Many, many more open source integrations

Couchbase embraces - and is embraced by - the open source community

Apache Drill	Arduino
Apache Flume	Puppet
Apache Storm	Ansible
Grunt	Talend
Hibernate	AppDynamics
Cloud Foundry	Nagios
Spring Security	Tomcat
CoreOS & AWS	Glassfish
Akka	J2EE

What we have?

Couchbase is widely supported and easily integrated

- ✓ **Big Data Analytics**: Apache Spark and Hadoop connectors
- ✓ **Big Data Streaming**: Apache Kafka connector
- ✓ **Full Text Search**: Apache Solr, CBFT (DP), Lucidworks, Elastic
- ✓ **Cloud Infrastructure**: AWS, Microsoft Azure, Google Cloud Platform
- ✓ **Cloud PAAS**: Joyent, Red Hat Openshift
- ✓ **Cloud Containers**: Docker
- ✓ **Business Intelligence**: ODBC/JDBC drivers
- ✓ **Enterprise Monitoring**: New Relic, AppDynamics
- ✓ **Enterprise Development**:
  Spring Data for Couchbase, Ottoman for Node.js, Linq2Couchbase for .NET

Couchbase

Learning Services