

# [What's New in Two]

## New features in Couchbase Server 2.0

With the introduction of Couchbase Server 2.0, Couchbase is transformed from a key-value store into a document database, adding support for JSON documents, indexing and querying, incremental map reduce and cross datacenter replication. Now Couchbase Server 2.0 can be used as both as a key-value store and a document database, leveraging the product's new flexible data model while building on its existing capabilities: easy scalability, high-performance and always-on characteristics.

The 2.0 release adds several new features making it easier for you to build richer and more powerful apps that suit many different use-cases including content and metadata management systems, social and mobile games, ad targeting platforms, user profiles stores, application session stores, chat and messaging platforms or high-availability caches. In this paper, we share more details about these features.

### What's new in Couchbase Server 2.0

#### Flexible data model

As a NoSQL document database, with Couchbase, there's no need to create and manage schemas. This greatly reduces the complexity and the time required to make changes to applications. Using documents, you can model your data exactly as it's represented in the application, as objects and entities. Plus, document lookups are fast as all related data is located together.

---

Couchbase Server 2.0 adds native support for JSON documents. Every JSON document can have a different record structure and multiple documents with different structures can be stored in the same Couchbase data bucket. Document structure can be changed at any time, without requiring modification to other documents in the database. This accelerates your ability to add features to your application.

The admin console (shown in Figure 1 and 2) has been extended to support creating, viewing, editing and deleting JSON documents. You can page through all the JSON documents in a Couchbase bucket, filter documents by key range or lookup a particular document by ID.

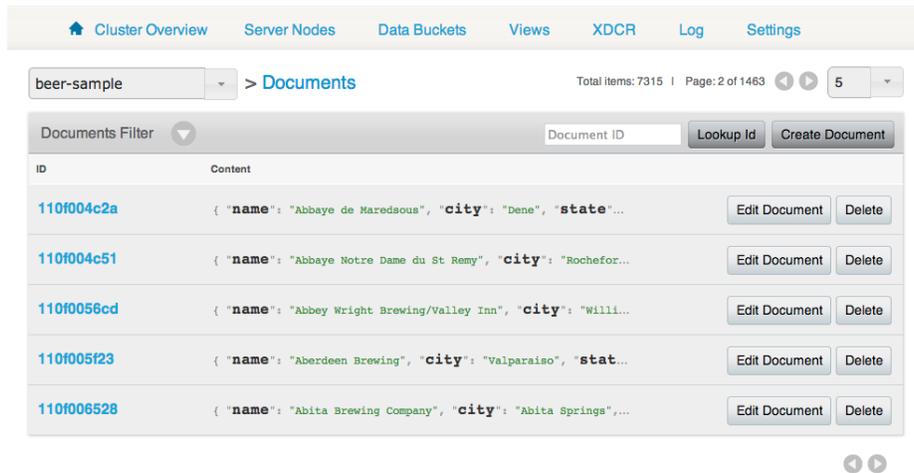


Figure 1: View JSON documents in the beer-sample bucket

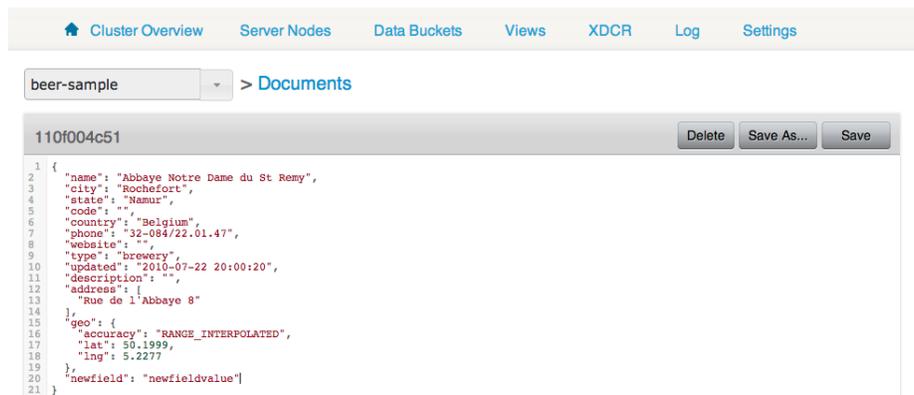


Figure 2: Editing a JSON document using the Couchbase Server admin console

You can read more about JSON support in Couchbase Server in the [JSON](#) section of the documentation.

## Indexing and querying

With Couchbase Server 2.0, you can easily index and query JSON documents. Indexes are defined using **design documents** and **views**. You create design documents on a per-bucket basis. Each design document can contain multiple views. Each view includes a **map** and optionally a **reduce** function, both written in Javascript. Once a view is defined, you can query data using this materialized view and lookup information based on various filters.

Views in Couchbase are built asynchronously and hence are **eventually indexed**. By default, queries on these views are **eventually consistent** with respect to document updates. However, if your application requires data to be indexed immediately, the SDKs offer an option to control this behavior on a per operation basis.

Couchbase distributes indexes across the cluster – each node indexing the data it holds. On each node, views are created on active documents and optionally on replica documents. During view building, the JavaScript map function is applied across every document on every node and the key value pairs for matching documents are retrieved and stored in the index. The index can be queried by the application using any of the Couchbase SDKs, including during rebalance.

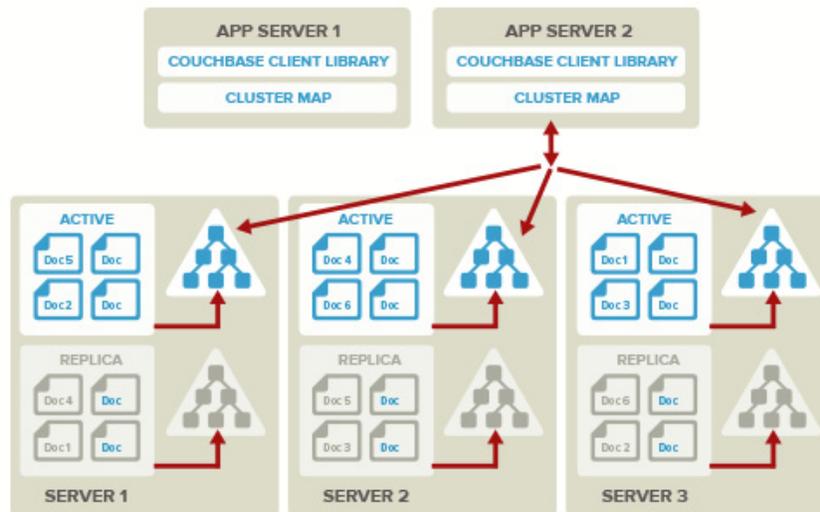


Figure 3: Distributed indexing and querying in Couchbase Server 2.0

Now let's take a look at indexing and querying across the cluster. The Couchbase SDK sends the query request to a randomly selected server within the cluster. The node that receives the query, distributes the request to other nodes in the cluster, then aggregates and returns the results to the client using a scatter and gather approach. For example, as shown in Figure 3, the Couchbase client library picks Server 2 and sends it a request. Using the index at each node, the query combines the results from each node as needed and sends the results back to server 2, which returns the results to the client.

Views can be used for a variety of use cases including primary index, secondary indexes, complex secondary indexes using composite keys, basic aggregation using built-in reduce functions, time-based rollups and simple real-time analytics such as leaderboards for tracking player performance in social games.

## Primary index

A primary index enables you to query document IDs over all the documents stored in the database and return matching documents. The view outputs the document ID of every document in the bucket using the document ID as the key.

```
VIEW CODE
Map
1 function (doc, meta) {
2   emit(meta.id, null);
3 }
4
```

Figure 4: Defining a primary index in Couchbase Server 2.0

## Secondary index

As shown in Figure 5, secondary indexes give developers the ability at view definition time, to decide which fields within the JSON documents they want to query. Like primary indexes, you can emit one or more attributes of the document as the key and query for an exact match or a range.

```
VIEW CODE
Map
1 function (doc, meta) {
2   if(doc.name)
3     emit(doc.name, null);
4 }
5
```

Figure 5: Defining a secondary index emitting a document attribute name

## Incremental map reduce

Couchbase's powerful JavaScript map reduce engine supports incremental index building, only re-indexing documents that have been changed since the last index update. Instead of recomputing view results from scratch every time the input changes, intermediate results are stored in the index and re-calculated only when necessary. This means no need to wait for a batch job across the entire data set to finish each time you want to see your results. You can read more about incremental map reduce functions in the [views chapter](#) of the documentation.

Incremental map reduce is very useful particularly when computing aggregates. There are three built-in reduce functions that can be pre-computed and stored in the index. This optimization means that you don't have to re-compute aggregates across the entire data set every time a query request an aggregate.

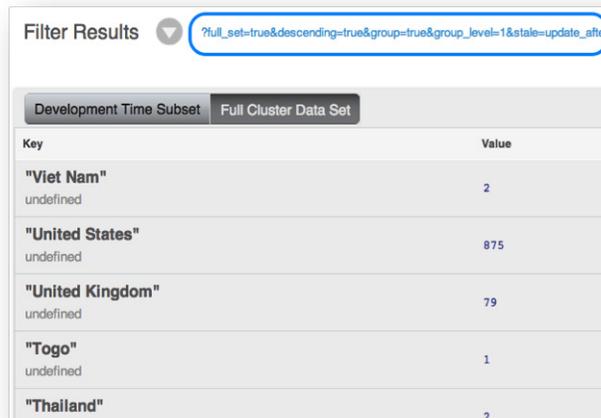


```
VIEW CODE

Map
1 function (doc, meta) {
2   .if(doc.country && doc.type == "brewery")
3   emit(doc.country, null);
4 }

Reduce (built in: _count, _sum, _stats)
1 _count
```

Figure 6: View in Couchbase Server with secondary index and built-in reduce



Filter Results

Development Time Subset Full Cluster Data Set

Key	Value
"Viet Nam" undefined	2
"United States" undefined	875
"United Kingdom" undefined	79
"Togo" undefined	1
"Thailand" undefined	2

Figure 7: Aggregated query results grouped by key in descending order

## Cross data center replication (XDCR)

While replication within a cluster helps with single node failures giving you high availability, your datacenter still remains at risk from catastrophic failures affecting all machines in your cluster – loss of power, natural disaster, etc.

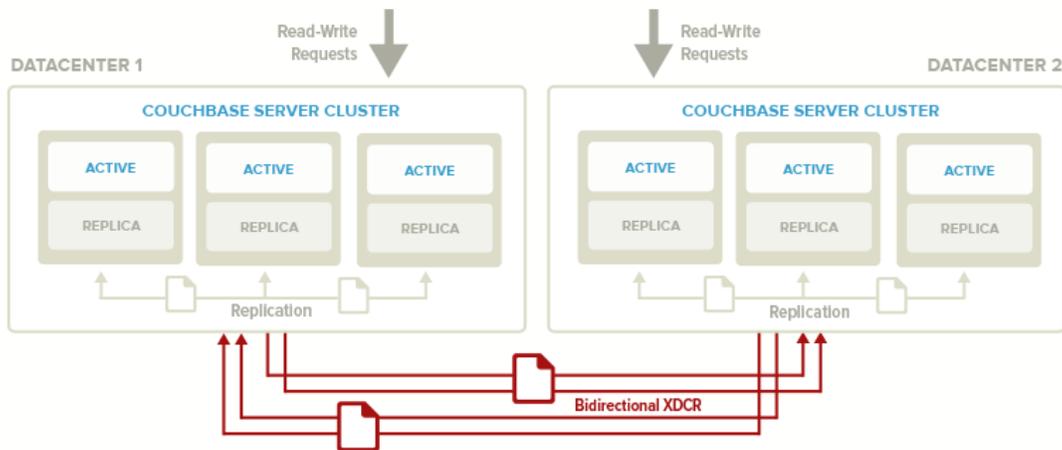


Figure 8: Cross datacenter replication across clusters

In 2.0, we add cross datacenter replication that allows you to replicate data from one cluster to the other so that your database is always on 24x365. It provides you with an easy way to replicate active data unidirectionally or bidirectionally to multiple datacenters across different geographical areas either for disaster recovery or to bring your data closer to your end users for reduced latencies.

XDCR and intra-cluster replication occur simultaneously. As shown in Figure 8, intra-cluster replication takes place within the clusters in both datacenters, while at the same time XDCR replicates documents across datacenters. Both datacenters are serving read and write requests from the application. It is a master-master topology across datacenters. XDCR is disk-based and data is replicated to the Datacenter 2 only after it is stored to disk on Datacenter 1 and vice versa.

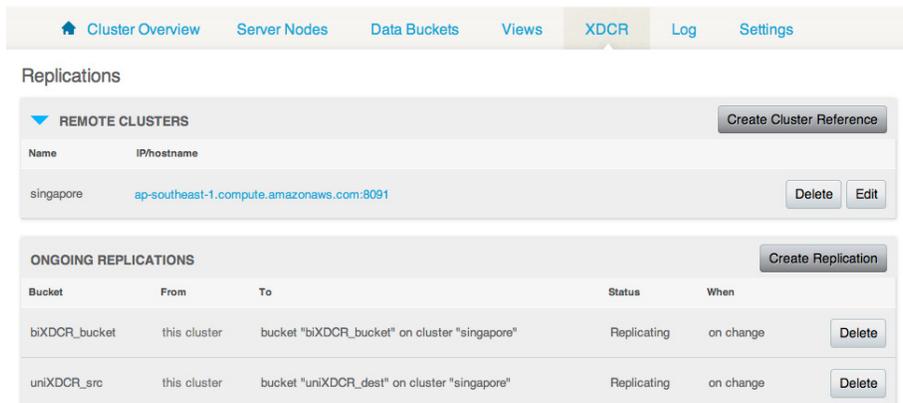


Figure 9: Couchbase Server admin console to configure XDCR

XDCR can easily be configured and managed through the Couchbase Server admin console (shown in Figure 9). Using the Couchbase admin console, all you need to do is point any node in the source cluster to any node in the target cluster. For bi-directional XDCR, do the same from the target cluster to the source cluster.

XDCR is network-failure resistant and will synchronize the data after network failure recovery. It is also topology aware. The clusters on both sides can have different number of nodes and can be scaled independently. In addition,

the enhanced UI provides stats for both the source and destination clusters. For more details, you might want to refer to the XDCR [documentation](#).

## Updated Couchbase SDKs

Couchbase Server 2.0 is accompanied by a variety of updated SDKs that include support for new features. For information on the new APIs, usages details and to download the SDKs, visit the [developer page](#) on our site.

## Additional improvements

In addition to the new features, Couchbase Server 2.0 includes several other improvements.

- **A new append-only persistence engine** that is immune to data corruption. This improves disk write performance as updates are written contiguously to the end of the file.
- **Online compaction** for data and indexes, which kicks off based on fragmentation thresholds. You can configure these thresholds along with the compaction schedule. In addition, the UI gives you more visibility into background compaction.
- **Improved working set management.** When your working set of documents becomes larger than allocated memory, the system needs to eject documents. Couchbase now tracks recently accessed, updated or inserted documents and ejects “not recently used” documents from RAM. This working set metadata is also replicated so that after a rebalance operation, your working set is maintained on the new nodes in the cluster for reduced latencies post rebalance.
- **Reduced server warm-up time.** When Couchbase Server is restarted, or when it is started after an offline recovery from backup, the server goes through a warm-up process first loading metadata from disk into RAM. Additionally, it loads the documents that belong to the tracked working set maintained in an access log. This reduces warm-up time and also makes the most relevant data available to the application after server restart.
- **New sample data buckets** are available to get you started with Couchbase Server 2.0. The buckets include sample documents as well as views to let you play around with the new features.

## About Couchbase Server

Couchbase Server NoSQL document-oriented database technology is optimized for the data management needs of interactive web and mobile applications. Web application developers and operators rely on Couchbase Server for:

- **Easy scalability:** It's easy to scale the data tier with Couchbase Server without any interruption or change to an application. With one click of a button, users can expand a cluster up to hundreds of servers, automatically keeping the workload evenly distributed among them.
- **Consistent high performance:** With consistent sub-millisecond response time, Couchbase Server helps ensure a positive experience for application users. Couchbase Server's high throughput means fewer servers are required to serve growing numbers of application users.
- **Always-on availability:** Couchbase Server's high availability features means applications are always online, 24x365. Whether upgrading the database, system software or hardware – or recovering from a disaster – Couchbase Server allows you to do so with zero downtime.
- **Flexible data model:** Couchbase Server leverages a JSON document model, eliminating the need to create and manage schemas, thereby dramatically reducing the complexity and time required when making changes to an application.

We can't wait to see the applications you build with Couchbase. Get started by [downloading](#) Couchbase Server 2.0.

### Useful links

- [Getting Started with Couchbase Server 2.0](#)
- [Couchbase Server 2.0 Manual](#)
- [Couchbase Server 2.0 Developer Guides](#)
- [Couchbase Server 2.0 SDKs](#)
- [Indexing and Querying in Couchbase Server 2.0](#)
- [Cross datacenter replication in Couchbase Server 2.0](#)
- [View writing best practices](#)