VIETNAM NATIONAL UNIVERSITY

UNIVERSITY OF INFORMATION TECHNOLOGY

FACULTY OF INFORMATION SYSTEM

DISTRIBUTED DATABASE MANAGEMENT SYSTEM
COUCHBASE SERVER

Phan Nguyễn Lâm Hà	21522030
Trương Gia Huy	21522172
Nguyễn Duy Khánh	21522208
Nguyễn Hoàng Sơn	21520433

Instructor: Mr. Nguyễn Hồ Duy Tri

Ho Chi Minh City, 2023

1. Introduction

1.1. Overview

Distributed Database Management System: CouchBase Server

- CouchBase Server is an open source, NoSQL, JSON document-oriented, distributed data platform. Data is stored as items, each having a key and a value.
- Sub-millisecond data operations are provided by powerful services for querying and indexing, and by a feature-rich, document-oriented query-language, SQL++
- Multiple instance can be combined intro a single cluster. A Cluster Manager program coordinates all node-activites.
- Data can be retained either in memory only, or in both memory and storage. Data can also be replicated across the nodes of the cluster to ensure that node-loss does not entail data-loss. Data items can also be selectively replicated across data center; for the purpose of either of backup only, or of stimulat-neous, multi-geo application-accress.

1.2. Authors, management organization and history

- Developers: CouchBase Inc
- Written in: C++, Erlang, C, Go, Java
- Intial release: August 2010
- Stable release: 7.2.2 / September 14,2023
- Membase was developed by several leaders of the memcached project, who had founded a company, NorthScale, to develop a key-value store with the simplicity, speed, and scalability of memcached, but also the storage, persistence and querying capabilities of a database. The original membase source code was contributed by NorthScale, and project co-sponsors Zynga and Naver Corporation (then known as NHN) to a new project on membase.org in June 2010.

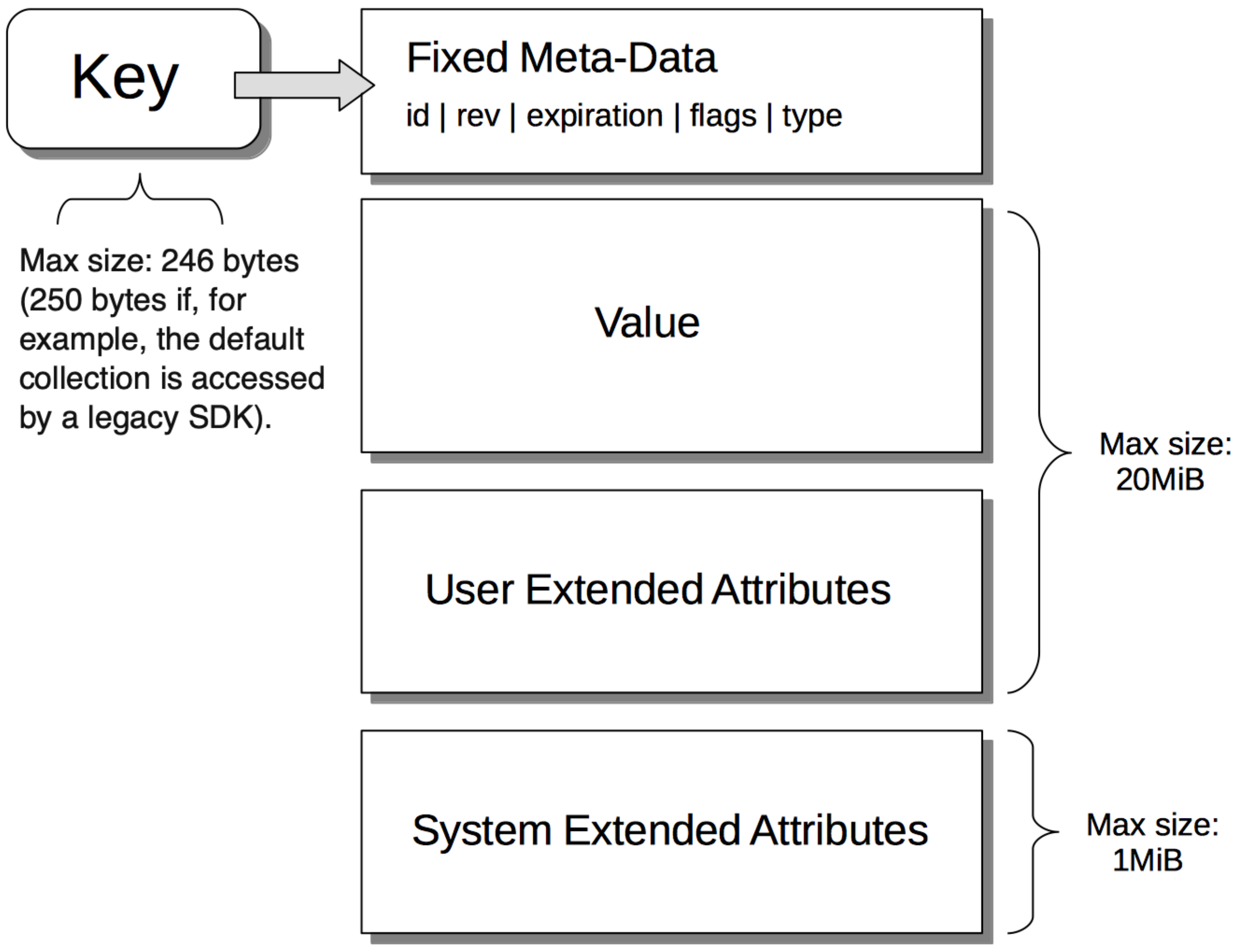
Couchbase product history at a glance

Table 1-1

2011	Merger of Membase and CouchOne to form Couchbase
2012	Couchbase Server 2.0 transforms Couchbase from a pure key-value database to a JSON document-oriented database
2014	Couchbase Mobile 1.0 – First mobile NoSQL database
2015	Couchbase Server 4.0 adds SQL-like queries with N1QL and Multi-Dimensional Scaling
2017	Couchbase Server 5.0 introduces Full-Text Search and ephemeral buckets
2018	Couchbase Server 6.0 eliminates ETL for operational workloads with the introduction of analytics based on SQL++; Couchbase Autonomous Operator for Kubernetes debuts in its 1.0 release
2019	Couchbase Server 6.5 adds distributed ACID transactions
2020	Couchbase debuts its Database-as-a-Service product, Couchbase Cloud™
2021	Couchbase Server 7.0 fuses the flexibility and scalability of NoSQL with the trusted strengths of relational database

1.3. Data Model

CouchBase Server saves data as items, each of which has a key and a value. Each key is unique within its bucket, and values can be either binary or JSON.


Key
Fixed Meta-Data
id | rev | expiration | flags | type
Max size: 246 bytes (250 bytes if, for example, the default collection is accessed by a legacy SDK).
Value
User Extended Attributes
Max size: 20MiB
System Extended Attributes
Max size: 1MiB

Picture 1-1

1.3.1. Keys

Each value is identified by a unique key(id), defined by the user or application when the items is saved. The key is immutable: once the items is saves, the key cannot be changed.

1.3.2. Values

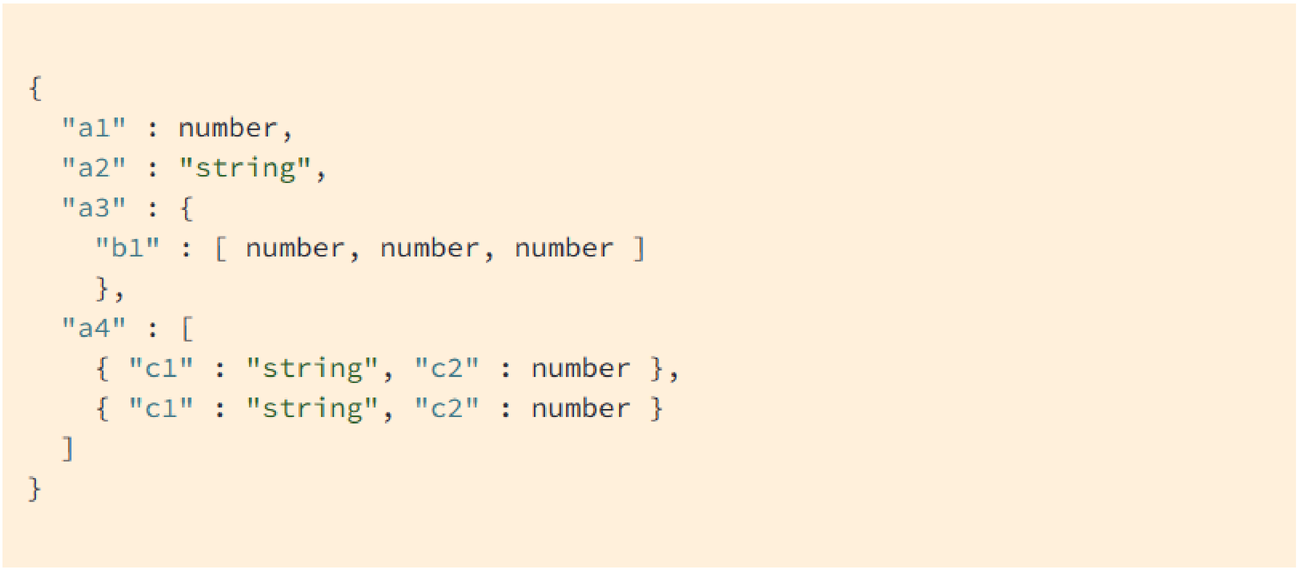
The maximum size of a value is 20 MiB. A value can be either:

- **Binary:** Any form of binary. Note that a binary value cannot be parsed, indexed, or queries: it can only be retrieved by key
- **JSON:** A JSON value, reffered to as a document, can be parsed, indexed and queries. Each document consist of one or more atrribute, each of which has its own value. An attribute's value can be a basic type, such as a number, string, or Boolean; or a complex, such as an embedded document or an array.

1.3.3. Document Structure

A sample JSON document is provided immediately below. Its contents are as follows:

- a1 and a2 are attributes that respectively have a single number and a single string as their values.
- a3 is an attribute whose value is an embedded document consisting of a single attribute, b1, whose own value is an array of three numbers.
- a4 is an attribute whose value is an array of two documents, each document consisting an attribute (c1 or c2) whose own values are respectively a string and a number.



```
{
  "a1" : number,
  "a2" : "string",
  "a3" : {
    "b1" : [ number, number, number ]
    },
  "a4" : [
    { "c1" : "string", "c2" : number },
    { "c1" : "string", "c2" : number }
  ]
}
```

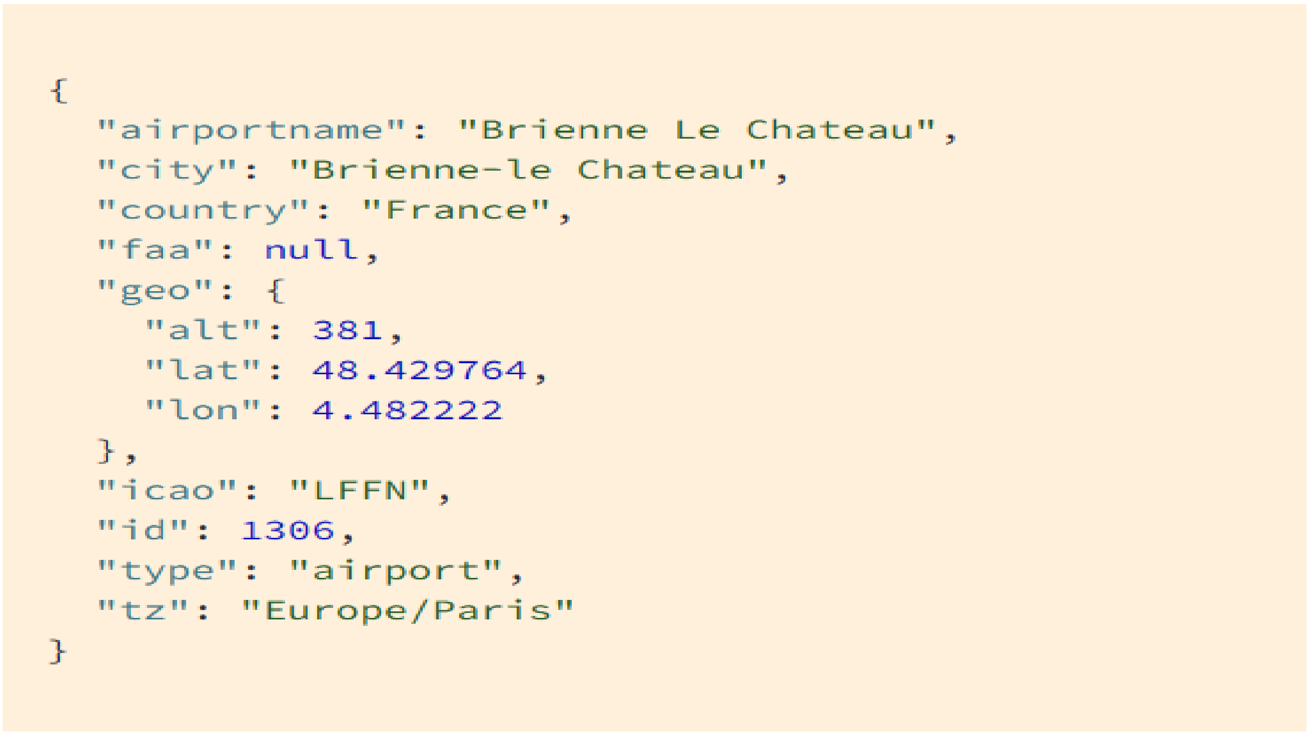
Picture 1-2

Sub-Document: an inner component of an JSON document. The sub-document is refferd to by a path, which specified attributes and array-position

Ex: a3.b1[0], value is the first number in an array

1.3.4. MetaData

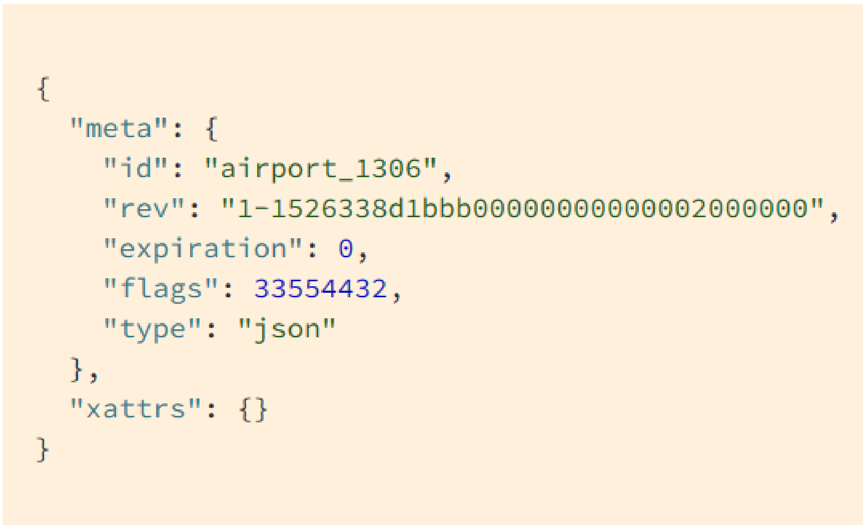
Metadata is automatically generated and stored for each item saved in Couchbase Server. For example, the following document, which contains airport-information, has been saved with the key **airport_1306:**



```
{
  "airportname": "Brienne Le Chateau",
  "city": "Brienne-le Chateau",
  "country": "France",
  "faa": null,
  "geo": {
    "alt": 381,
    "lat": 48.429764,
    "lon": 4.482222
  },
  "icao": "LFFN",
  "id": 1306,
  "type": "airport",
  "tz": "Europe/Paris"
}
```

Picture 1-3

The meta data might be as follows:



```
{
  "meta": {
    "id": "airport_1306",
    "rev": "1-1526338d1bbb0000000000000002000000",
    "expiration": 0,
    "flags": 33554432,
    "type": "json"
  },
  "xattrs": {}
}
```

Picture 1-4

The attributes within the metadata are:

- meta: The attribute whose value is the standard metadata for the saved document, airport_1306.
- **id: The key of the saved document, which is airport_1306.**
- rev: The revision or sequence number. This value is for internal server-use only: it is used in the resolution of conflicts that occur when replicated documents are

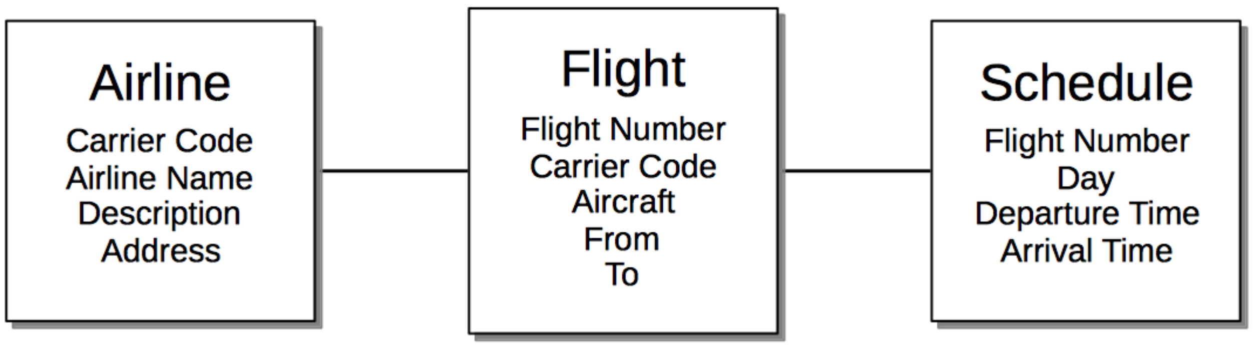
updated concurrently on different servers, representing the number of times the document has been mutated.

- expiration: The expiration-time (or Time-To-Live) of the document. If non-zero, this determines the point at which the document is removed from the system.
- flags: Couchbase SDK-specific values that may be used to identify the type of data saved, or to specify formatting.
- type: The type of the saved value, which in this case is json.
- xattrs: Extended Attributes, which constitute a special kind of metadata, some of which is system-internal, some of which can optionally be written and read by user-applications.

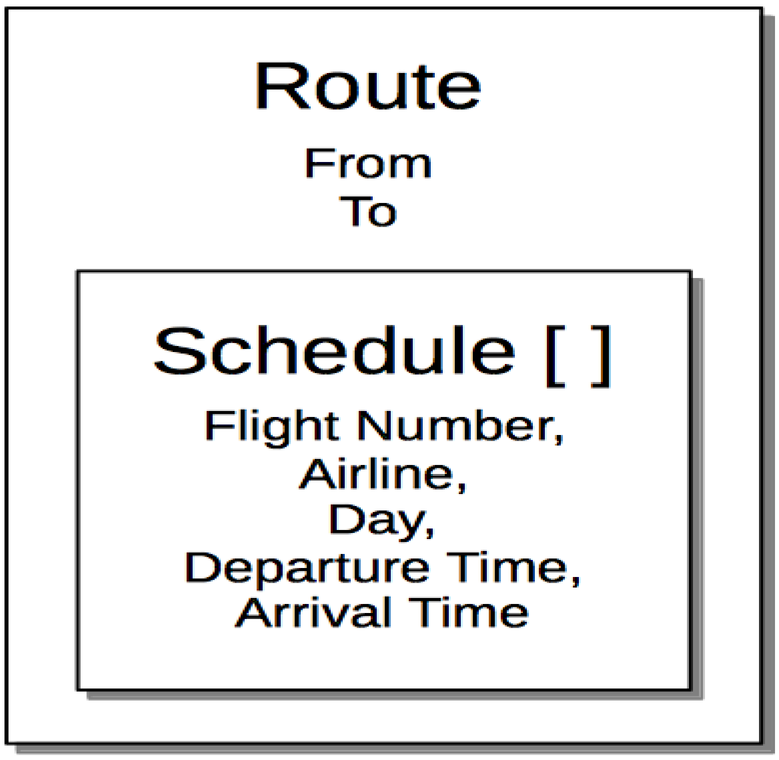
1.3.5. Document vs Relational Model

- The CouchBase data model is based on JSON, which provides a simple, lightweight, human-readable notation.
- An individual document often represents a single instance of an object in application code. A document might be considered equivalent to a row in a relational table; with each of the document’s attributes being equivalent to a column. **Couchbase, however, provides greater flexibility than relational databases, in that it can store JSON documents with varied schemas.**
- Documents can contain nested structures. This allows developers to express many-to-many relationships without requiring a reference or junction table; and is naturally expressive of hierarchical data.

Example: The nature and value of the document data model is clarified by comparison with the relational. To support an online flight-booking application, allowing users to search for flights by date, the relational model requires multiple tables — for flights, airlines, and schedules. The result may be as follows:



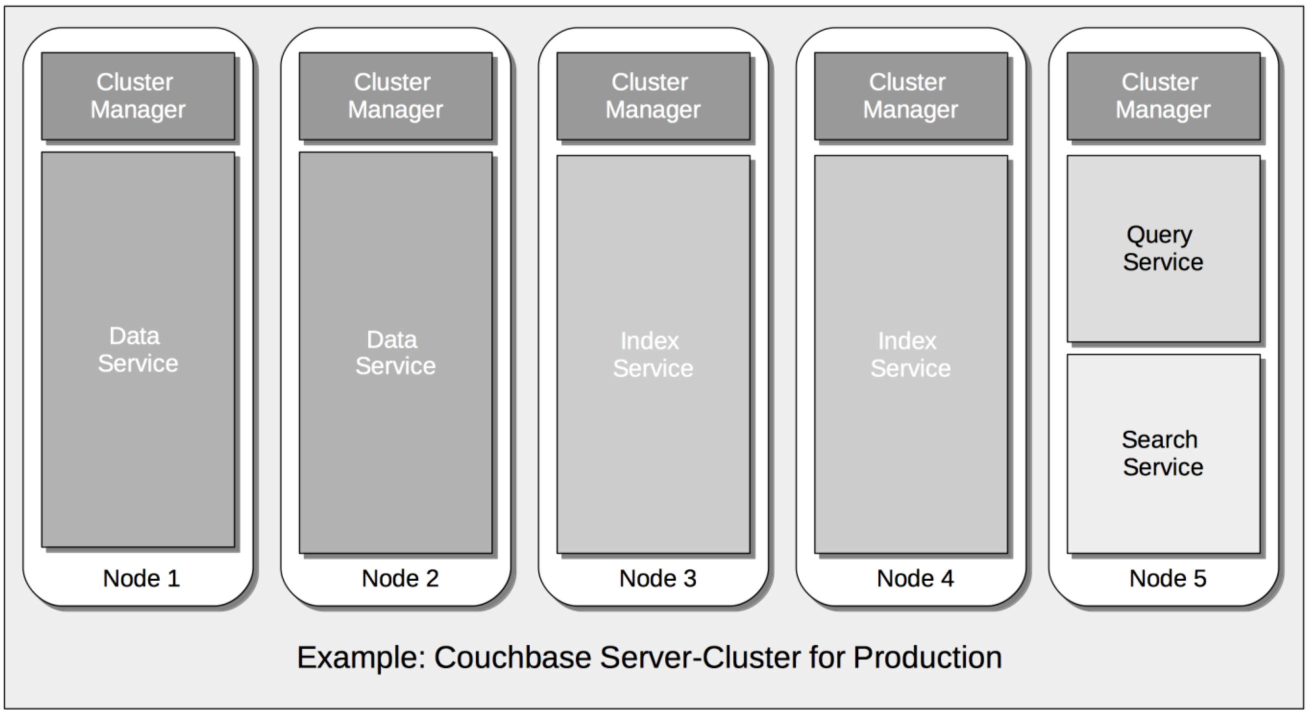
By contrast, the document model likely requires only a single document, which embeds an array of schedules for all flights between each of two airports:



Thus, in the document model, each document can be highly self-contained. This supports the rapid fulfillment of application-requests, and has important implications for both scalability and latency: one document can be replicated, or atomically changed, without other documents needing to be accessed; eradicating the need for complex inter-node coordination, and minimizing contention

1.4. Services

- Couchbase Services support access to and maintenance of data. Services can be deployed with flexibility across available hardware-resources, providing Multi-Dimensional Scaling, whereby a cluster can be tuned for optimal handling of emergent workloads.
- Services are configured and deployed by the Full Administrator who initializes Couchbase Server on one or more nodes. The standard configuration-sequence allows a subset of services to be selected per node, with an individual memory-allocation for each. Each service supports a particular form of data-access. Services not required need not be deployed. Services intended to support a heavy workload can be deployed across multiple cluster-nodes, to ensure optimal performance and resource-availability.



Picture 1-5

In this revised configuration, the Data Service is the only service to run on two of the nodes; the Index Service the only service on two futher nodes; and the Query and Search Services share the fifth and final node.

Couchbase Server provides the following services:

- **Data**: Supports the storing, setting, and retrieving of data-items, specified by key.
- **Query**: Parses queries specified in the *N1QL* query-language, executes the queries, and returns results. The Query Service interacts with both the Data and Index services.
- **Index**: Creates indexes, for use by the Query and Analytics services.
- **Search**: Create indexes specially purposed for *Full Text Search*. This supports language-aware searching; allowing users to search for, say, the word beauties, and additionally obtain results for beauty and beautiful.
- **Analytics**: Supports join, set, aggregation, and grouping operations; which are expected to be large, long-running, and highly consumptive of memory and CPU resources.
- **Eventing**: Supports near real-time handling of changes to data: code can be executed both in response to document-mutations, and as scheduled by timers.
- **Backup**: Supports the scheduling of full and incremental data backups, either for specific individual buckets, or for all buckets on the cluster. Also allows the scheduling of *merges* of previously made backups.

- These services can be deployed, maintained, and provisioned independently of one another, by means of *Multi-Dimensional Scaling*, to ensure the most effective ongoing response to changing business conditions and emergent workload-requirements

1.5. Bucket

A bucket is the fundametal space for storing data in CouchBase Server. Each bucket contain a hierachy of scope and collection to group keys and values logically.

1.5.1. Bucket Types

A maximum of 30 buckets can be created in a cluster. Each bucket must be specified as one of the following three types.

- **Couchbase buckets:** These store data persistently, as well as in memory. They allow data to be automatically replicated for high availability, using the Database

Change Protocol (DCP); and dynamically scaled across multiple clusters, by means of Cross Datacenter Replication (XDCR).

If a Couchbase bucket's RAM-quota is exceeded, items are ejected. This means that data, which is resident both in memory and on disk, is removed from memory, but not from disk. Therefore, if removed data is subsequently needed, it is reloaded into memory from disk. For a Couchbase bucket, ejection can be either of the following, based on configuration performed at the time of bucket-creation:

- Value-only: Only key-values are removed. Generally, this favors performance at the expense of memory.

- Full: All data — including keys, key-values, and metadata — is removed. Generally, this favors memory at the expense of performance.

- **Ephemeral buckets:** These are an alternative to Couchbase buckets, to be used whenever persistence is not required: for example, when repeated disk-access involves too much overhead. This allows highly consistent in-memory performance, without disk-based fluctuations. It also allows faster node rebalances and restarts.

If an Ephemeral bucket's RAM-quota is exceeded, one of the following occurs, based on configuration performed at the time of bucket-creation:

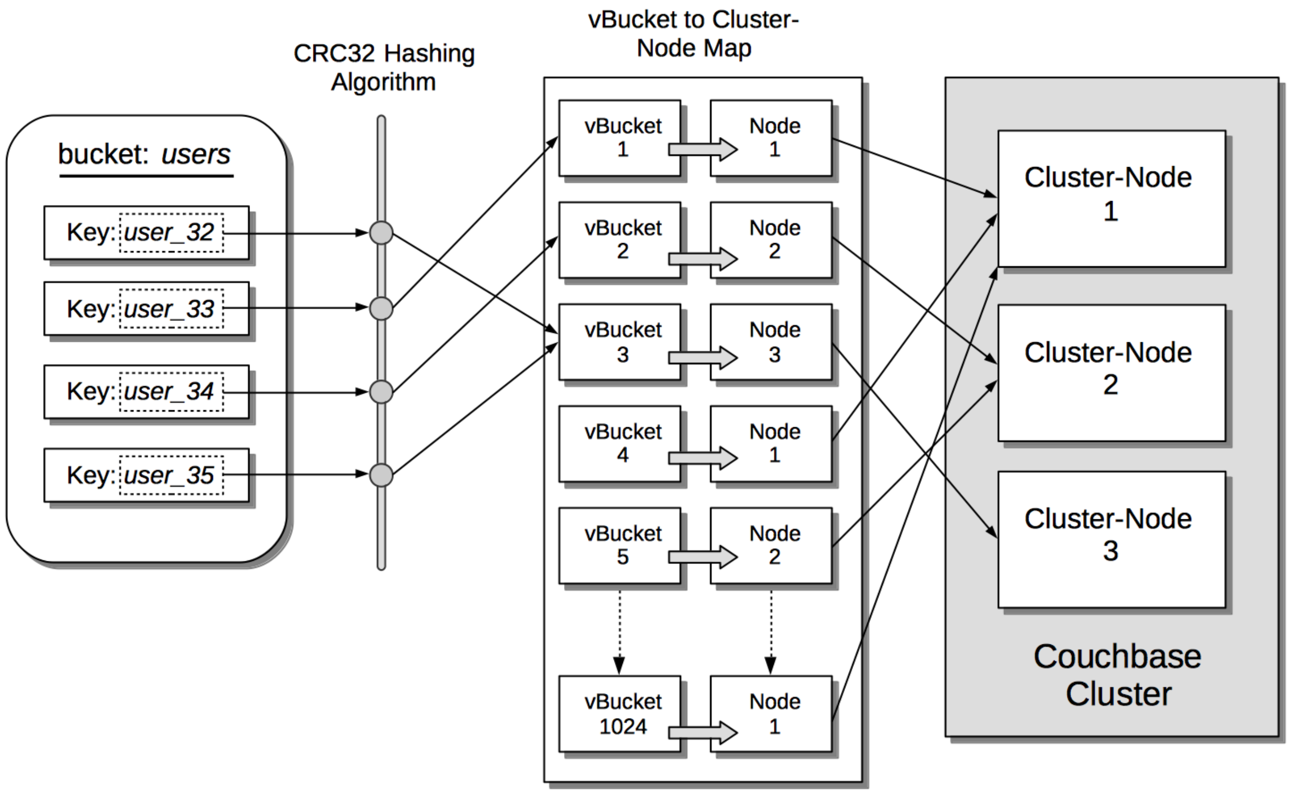
- Resident data-items remain in RAM. No additional data can be added; and attempts to add data therefore fail.

- Resident data-items are ejected from RAM, to make way for new data. For an Ephemeral bucket, this means that data, which is resident in memory (but, due to this type of bucket, can never be on disk), is removed from memory. Therefore, if removed data is subsequently needed, it cannot be re-acquired from Couchbase Server.

1.5.2. Vbuckets

vBuckets are virtual buckets that help distribute data effectively across a cluster, support replication across multiple nodes.

- Both CouchBase and Ephermal buckets are implemented as Vbuckets, up to 1024 of which are created for every bucket.
- Buckets's items are distributed evenly across its vBukets
- The 1024 vBuckets that implement a defined bucket are referred to as active vBuckets. If a bucket is replicated, each replica is implemented as a further 1024 vBuckets, referred to as replica vBuckets. Write operations are performed only on active vBuckets. Most read operations are performed on active vBuckets, though items can also be read from replica vBuckets when necessary.
- Items are written to and retrieved from vBuckets by means of a CRC32 hashing algorithm, which is applied to the item's key, and so produces the number of the vBucket in which the item resides. vBuckets are mapped to individual nodes by the Cluster Manager: the mapping is constantly updated and made generally available to SDK and other clients.
- The relationships between a bucket, its keys (and their associated values), the hashing algorithm, vBuckets, server-mappings, and servers, is illustrated below:



Picture 1-6

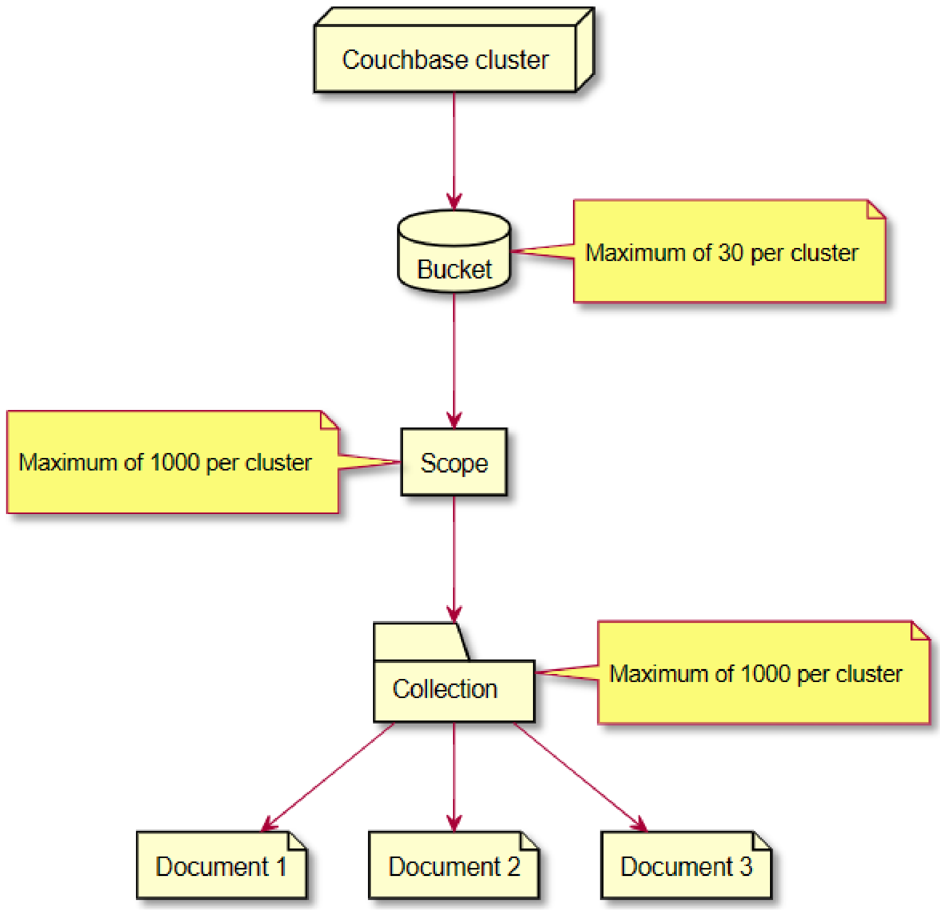
Thus, an authorized client attempting to access data performs a hash operation on the appropriate key, and thereby calculates the number of the vBucket that owns the key. The client then examines the vBucket map to determine the server-node on which the vBucket resides; and finally performs its operation directly on that server-node.

- In Couchbase Server Version 7.0+, documents within a bucket are organized into Scopes and Collections. Scopes and collections do not affect the way in which keys are allocated to vBuckets.

1.5.3. Scope and Collections

- Collection: A data container, within a bucket whose type is either CouchBase or Ephemeral.
  - Item-names must be unique within their collections
  - Items can optionally be assigned to different collections according to content-type
- Scope: a mechanism for grouping of multiple collections.
  - Collection-names must be unique within their scope

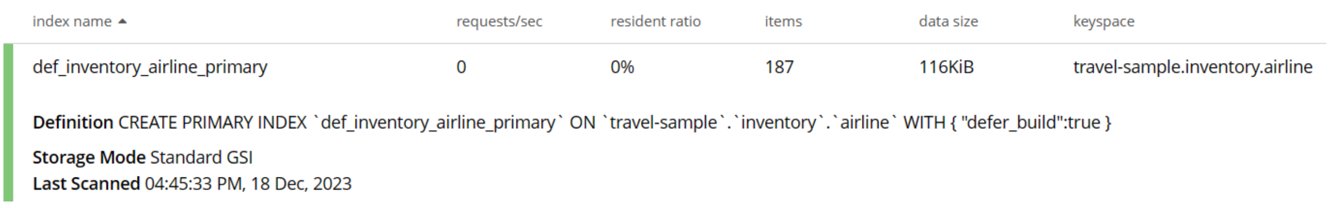
- Collection might be assigned to different scope according to content-type, or deployment-phase


Couchbase cluster
Bucket
Maximum of 30 per cluster
Maximum of 1000 per cluster
Scope
Collection
Maximum of 1000 per cluster
Document 1
Document 2
Document 3

Picture 1-7

2.2.1. Indexes

- Indexes enhance the performance of querry and search operation
- Indexes are used by certain services, such as Query, Analytics, and Search, as targets for search-routines. Each index makes a predefined subset of data available for the search.
- The Query service relies on indexes provided by the Index service. The Search and Analytics services both provide their own indexes, internally.
- The following forms of index are available:
    - Primary: Provided by the Index Service, this is based on the unique key of every item in a specified collection. Every primary index is maintained asynchronously. A primary index is intended to be used for simple queries, which have no filters or predicates.

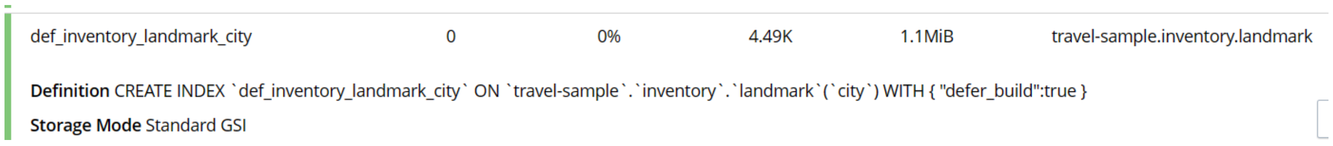


index name ▲	requests/sec	resident ratio	items	data size	keyspace
def_inventory_airline_primary	0	0%	187	116KiB	travel-sample.inventory.airline

Definition CREATE PRIMARY INDEX `def_inventory_airline_primary` ON `travel-sample`.`inventory`.`airline` WITH { "defer_build":true }
Storage Mode Standard GSI
Last Scanned 04:45:33 PM, 18 Dec, 2023

Picture 1-8

- Secondary: Provided by the Index Service, this is based on an attribute within a document. The value associated with the attribute can be of any type: scalar, object, or array. A Secondary Index is frequently referred to as a Global Secondary Index, or GSI. This is the kind of index used most frequently in Couchbase Server, for queries performed with SQL++



def_inventory_landmark_city	0	0%	4.49K	1.1MiB	travel-sample.inventory.landmark

Definition CREATE INDEX `def_inventory_landmark_city` ON `travel-sample`.`inventory`.`landmark`(`city`) WITH { "defer_build":true }
Storage Mode Standard GSI

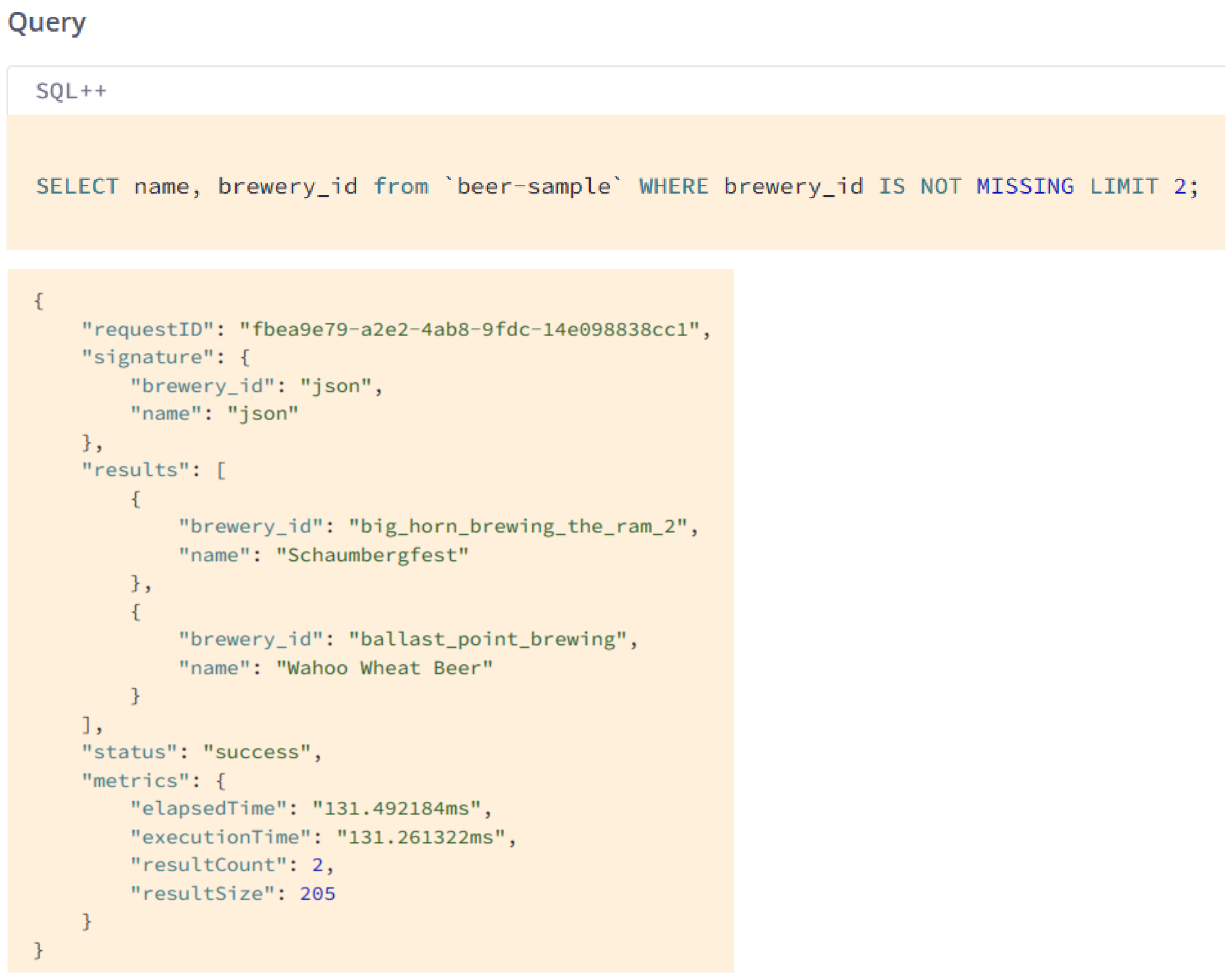
Picture 1-9

- Full Text: Provided by the Search Service, this is a specially purposed index, which contains targets derived from the textual contents of documents within one or more specified keyspaces. Text-matches of different degrees of exactitude can be searched for. Both input and target text-values can be purged of irrelevant characters (such as punctuation marks or html tags).

1.6. SQL++ Language

- CouchBase Server can be queried using SQL++.
- SQL++ is an expressive, powerful, and complete SQL dialect for querying, transforming, and manipulating JSON data. Based on SQL, it’s immediately familiar to developers who can quickly start developing rich applications.
- The SQL++ language is composed of statements, expressions and comments
  - Statement:
    - Data Definition Language(DDL) statement to create indexes, modify indexes and drop indexes
    - Data Manipulation Language(DML) statement to select,insert,update,delete, and upsert data into JSON documents

- Expression:
    - Literal value
    - Identifiers
    - Arithmetic terms
    - Comparison terms
    - Concatenation terms
    - Logical terms
    - Conditional expressions
    - Collection expressions
    - Construction expressions
    - Nested expressions
    - Function calls
    - Subqueries
- Comments: block comments and line comments



Query

SQL++

```
SELECT name, brewery_id from `beer-sample` WHERE brewery_id IS NOT MISSING LIMIT 2;
```

```
{
    "requestID": "fbea9e79-a2e2-4ab8-9fdc-14e098838cc1",
    "signature": {
        "brewery_id": "json",
        "name": "json"
    },
    "results": [
        {
            "brewery_id": "big_horn_brewing_the_ram_2",
            "name": "Schaumbergfest"
        },
        {
            "brewery_id": "ballast_point_brewing",
            "name": "Wahoo Wheat Beer"
        }
    ],
    "status": "success",
    "metrics": {
        "elapsedTime": "131.492184ms",
        "executionTime": "131.261322ms",
        "resultCount": 2,
        "resultSize": 205
    }
}
```

Picture 1-10

1.7. Local and Remote Replication

CouchBase Server ensure the availability of data across the node of a cluster; across group of nodes within the cluster; and across separate clusters, potentially located in different data-centers.

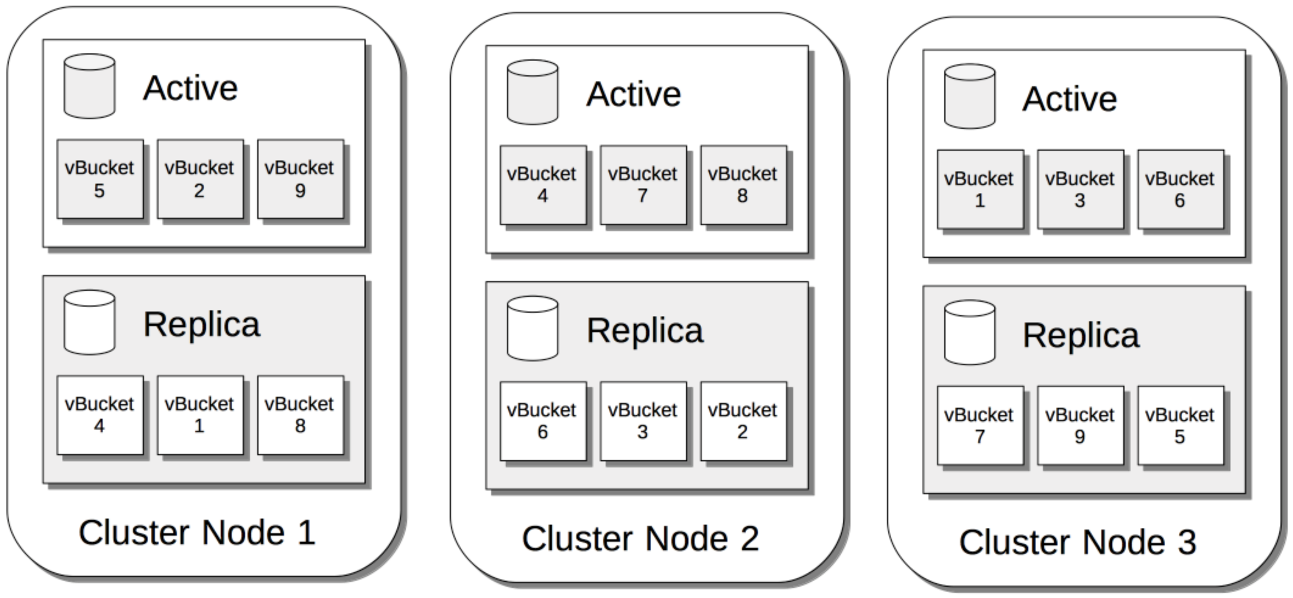
- Local, or intra-cluster replication
- Remote, or Cross Data Center Replication (XDCR)

1.7.1. Intra-Cluster Replication

Intra-cluster replication involves replicating data across the nodes of a cluster.

Couchbase Data is defined logically to reside in Buckets. Each bucket, when defined, is implemented by Couchbase Server as vBuckets, up to 1024 of which thereby exist in memory (and, in the case of Couchbase Buckets, on disk); the exact number at any given time depending on the number of items to be stored. Items are associated with vBuckets by means of a hashing algorithm, and buckets are assigned to nodes according to a fixed mapping, determined and updated by the Master Services of the ns-server component of the Cluster Manager.

Up to three replica buckets can be defined for every bucket. Each replica itself is also implemented as 1024 vBuckets. A vBucket that is part of the original implementation of the defined bucket referred to as an active vBucket. Therefore, a bucket defined with two replicas has 1024 active vBuckets and 2048 replica vBuckets. Typically, only active vBuckets are accessed for read and write operations: although replica vBuckets are able to support read requests. Nevertheless, vBuckets receive a continuous stream of mutations from the active vBucket by means of the Database Change Protocol (DCP), and are thereby kept constantly up to date. To ensure maximum availability of data in case of node-failures, the Master Services for the cluster calculate and implement the optimal vBucket Distribution across available nodes: consequently, the chance of data-loss through the failure of an individual node is minimized, since replicas are available on the nodes that remain. This is shown by the following illustration:



Picture 1-11

The illustration shows active and replica vBuckets that correspond to a single, user-defined bucket, for which a single replication-instance has been specified. The first nine active vBuckets are shown, along with their nine corresponding, replica vBuckets; distributed across three server-nodes. The distribution of vBuckets indicates a likely distribution calculated by Couchbase Server: no replica resides on the same node as its active equivalent: therefore, should any one of the three nodes fail, its data remains available.

1.7.2. Cross Data Center Replication(XDCR)

- Cross Data Center Replication (XDCR) replicates data between a source bucket and a target bucket. The buckets may be located on different clusters, and in different data centers: this provides protection against data-center failure, and also provides high-performance data-access for globally distributed, mission-critical applications.
- Data from the source bucket is pushed to the target bucket by means of an XDCR agent, running on the source cluster, using the Database Change Protocol. Any bucket (Couchbase or Ephemeral) on any cluster can be specified as a source or a target for one or more XDCR definitions.

1.7.2.1. Different between XDCR and ICR

- Cross Data Center Replication differs from intra-cluster replication in the following, principal ways:
  - As indicated by their respective names, intra-cluster replication replicates data across the nodes of a single cluster; while Cross Data Center Replication replicates data across multiple clusters, each potentially in a different data center.
  - Whereas intra-cluster replication is configured and performed with reference to only a single bucket (to which all active and replica vBuckets will correspond), XDCR requires two buckets to be administrator-specified, for a replication to occur: one is the bucket on the source cluster, which provides the data to be replicated; the other is the bucket on the target cluster, which receives the replicated data.

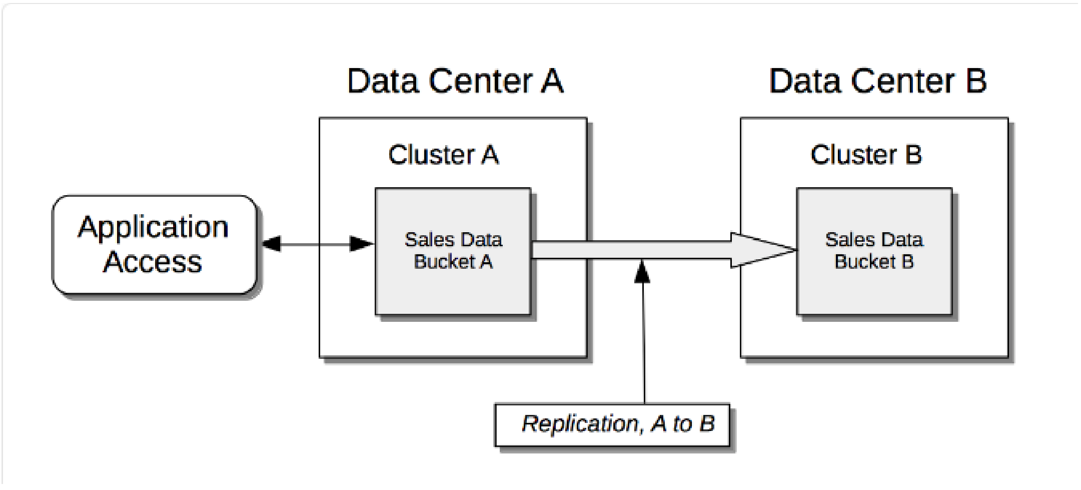
- Whereas intra-cluster replication is configured at bucket-creation, XDCR is configured following the creation of both the source and target buckets.

- The starting, stopping, and pausing of XDCR all occur independently of whatever intra-cluster replication is in progress on either the source or target cluster. While running, XDCR continuously propagates mutations from the source to the target bucket.

1.7.2.2. XDCR Direction and Topology

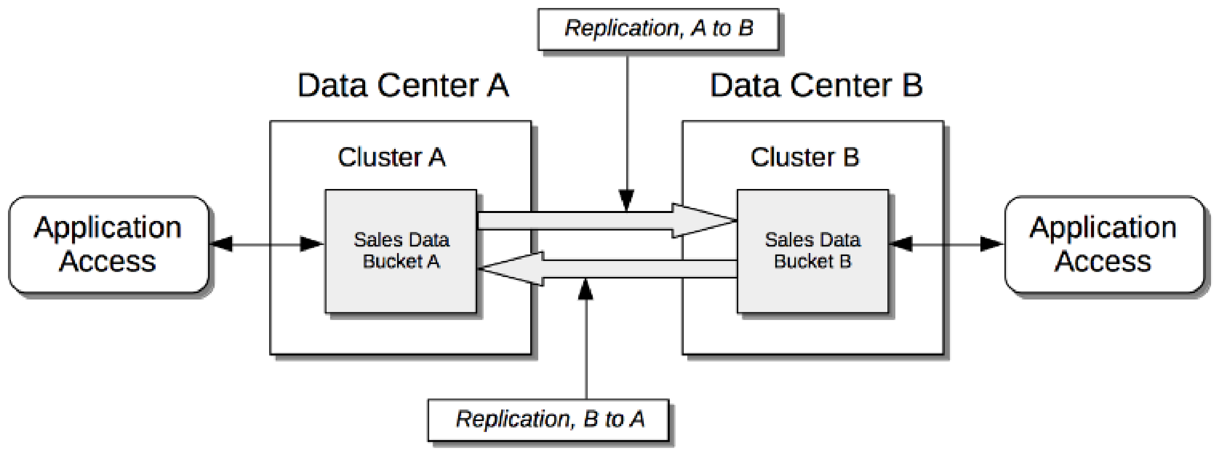
XDCR allows replication to occur between source and target clusters in either of the following ways:

- **Unidirectionally:** The data contained in a specified source bucket is replicated to a specified target bucket. Although the replicated data on the source could be used for the routine serving of data, it is in fact intended principally as a backup, to support disaster recovery.



Picture 1-12

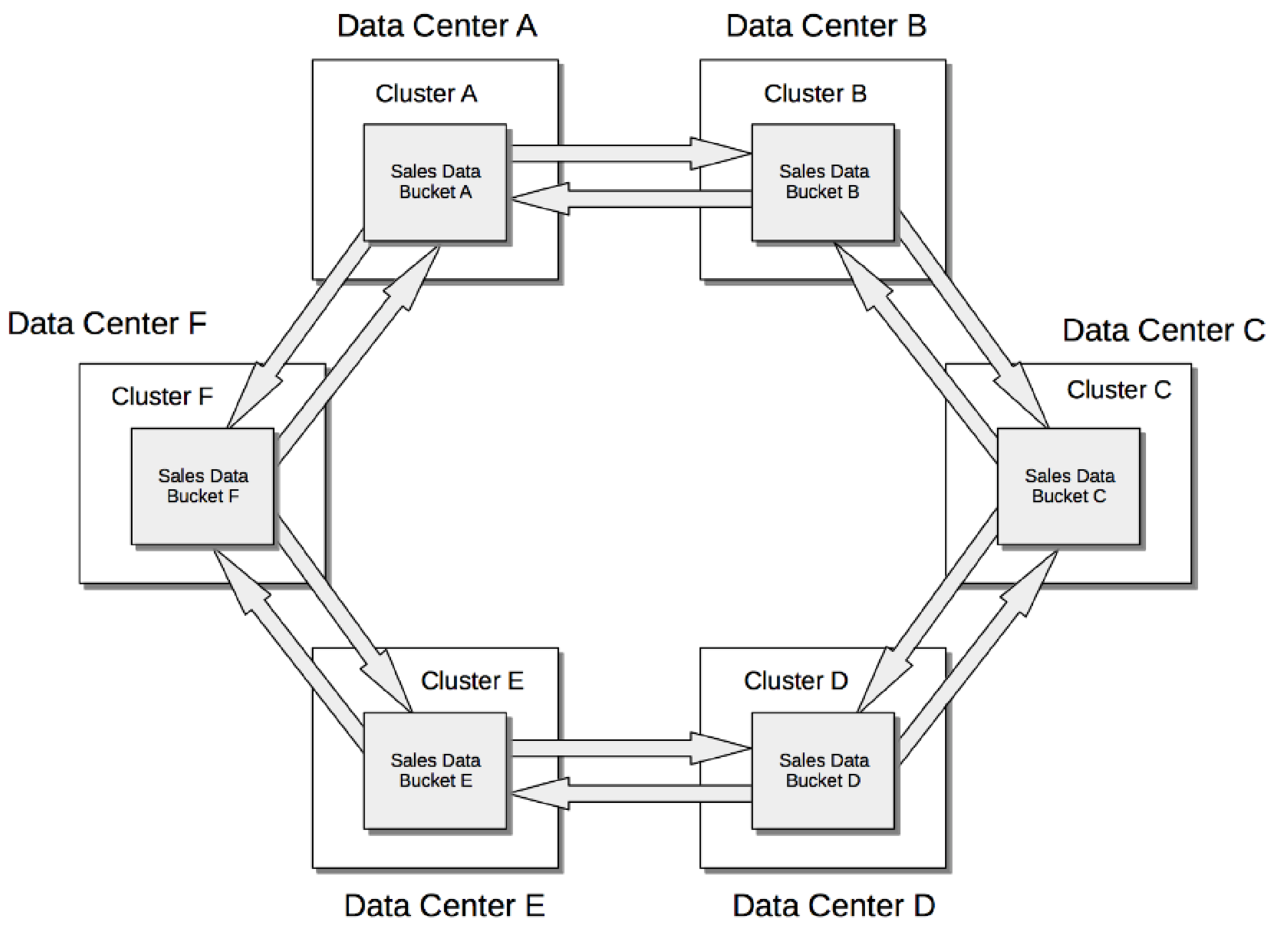
- **Bidirectionally:** The data contained in a specified source bucket is replicated to a specified target bucket; and the data contained in the target bucket is, in turn, replicated back to the source bucket. This allows both buckets to be used for the serving of data, which may provide faster data-access for users and applications in remote geographies.



Picture 1-13

Note that XDCR provides only a single basic mechanism from which replications are built: this is the unidirectional replication. A bidirectional topology is created by implementing two unidirectional replications, in opposite directions, between two clusters; such that a bucket on each cluster functions as both source and target.

Used in different combinations, unidirectional and bidirectional replication can support complex topologies; an example being the ring topology, where multiple clusters each connect to exactly two peers, so that a complete ring of connections is formed:



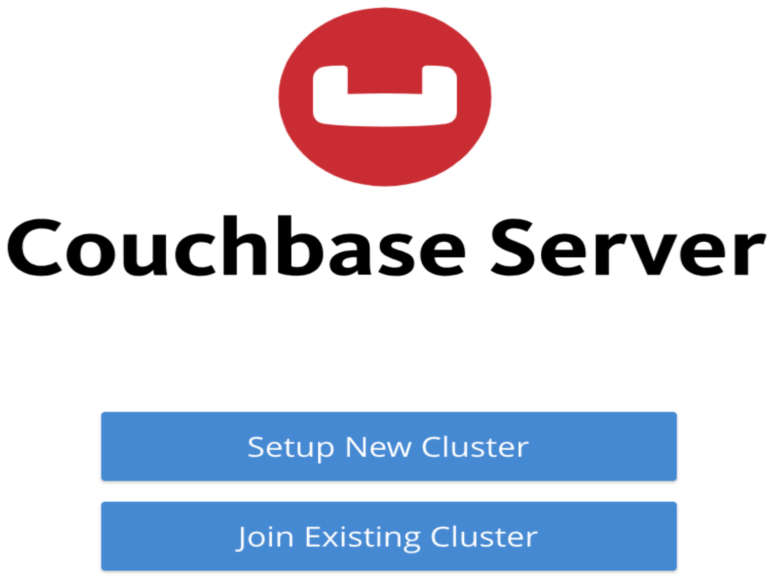
Picture 1-14

2. INSTALLATION AND SET UP

2.1. Installation on one node

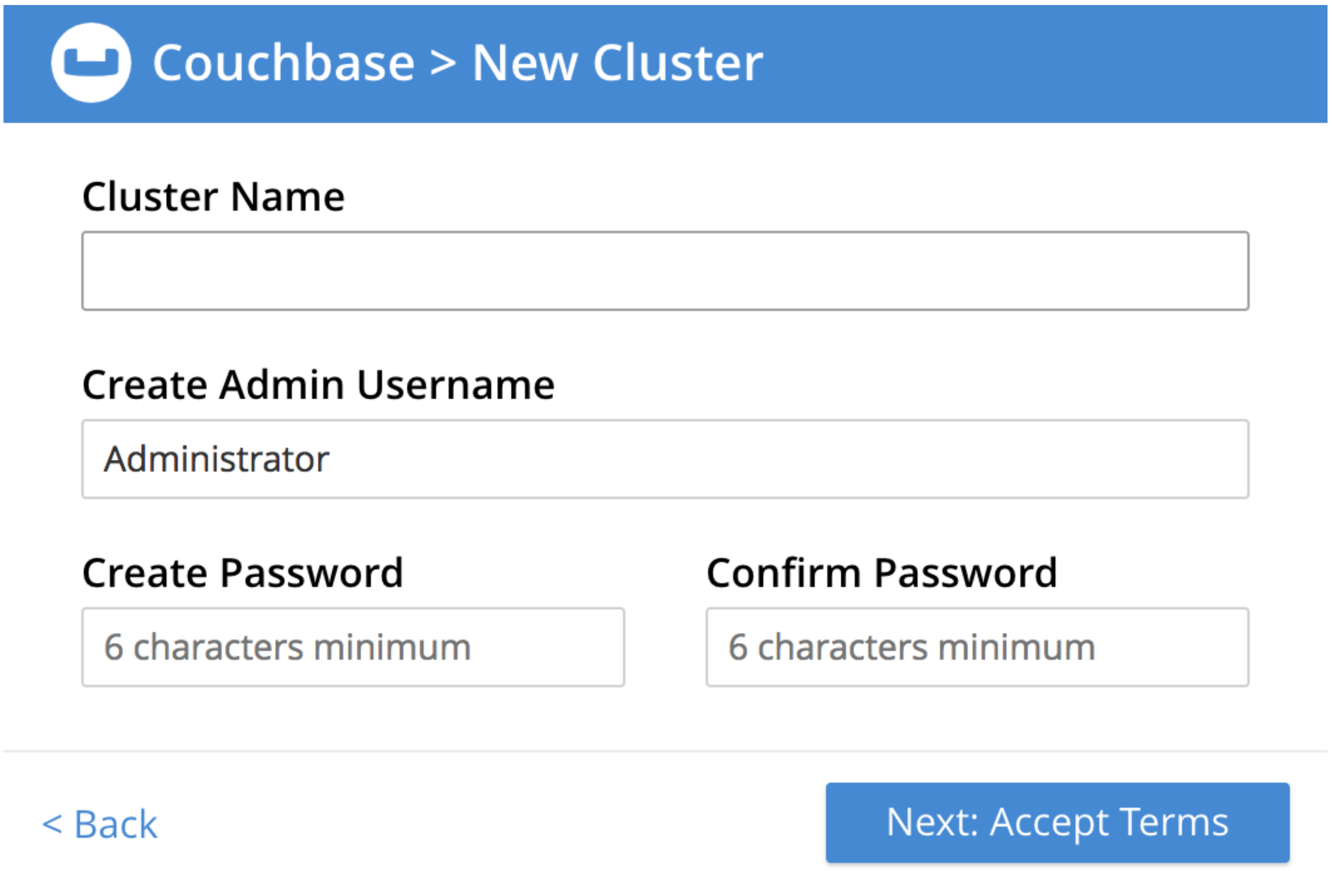
2.1.1. Create a Cluster

- Connect to CouchBase Web Console via: **http://localhost:8091**



Picture 2-1

- Click on: Setup New Cluster



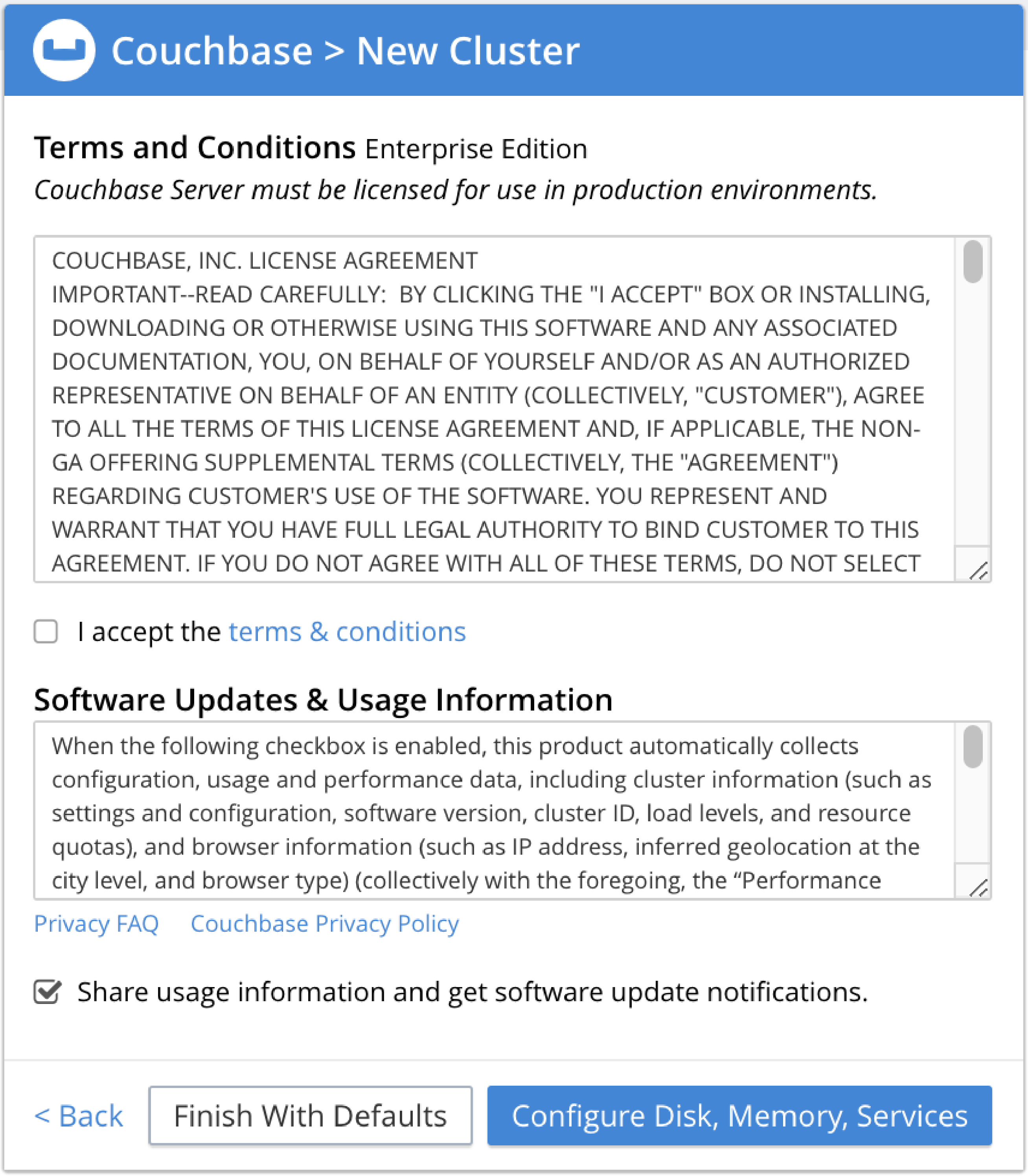
Picture 2-2

The field display on the screen are:

- Cluster Name: Name for the cluster to be created

- Create Admin Username: your choice of username, Full Administrator: read-write access to all CouchBase Server resources, including the ability to create new users with defined roles and corresponding privileges.
- Create Password: choice of password

- When you have enter appropriate data, left click on Next: Accept Term



Couchbase > New Cluster

Terms and Conditions Enterprise Edition

Couchbase Server must be licensed for use in production environments.

COUCHBASE, INC. LICENSE AGREEMENT
IMPORTANT--READ CAREFULLY: BY CLICKING THE "I ACCEPT" BOX OR INSTALLING, DOWNLOADING OR OTHERWISE USING THIS SOFTWARE AND ANY ASSOCIATED DOCUMENTATION, YOU, ON BEHALF OF YOURSELF AND/OR AS AN AUTHORIZED REPRESENTATIVE ON BEHALF OF AN ENTITY (COLLECTIVELY, "CUSTOMER"), AGREE TO ALL THE TERMS OF THIS LICENSE AGREEMENT AND, IF APPLICABLE, THE NON-GA OFFERING SUPPLEMENTAL TERMS (COLLECTIVELY, THE "AGREEMENT") REGARDING CUSTOMER'S USE OF THE SOFTWARE. YOU REPRESENT AND WARRANT THAT YOU HAVE FULL LEGAL AUTHORITY TO BIND CUSTOMER TO THIS AGREEMENT. IF YOU DO NOT AGREE WITH ALL OF THESE TERMS, DO NOT SELECT

☐ I accept the terms & conditions

Software Updates & Usage Information

When the following checkbox is enabled, this product automatically collects configuration, usage and performance data, including cluster information (such as settings and configuration, software version, cluster ID, load levels, and resource quotas), and browser information (such as IP address, inferred geolocation at the city level, and browser type) (collectively with the foregoing, the “Performance

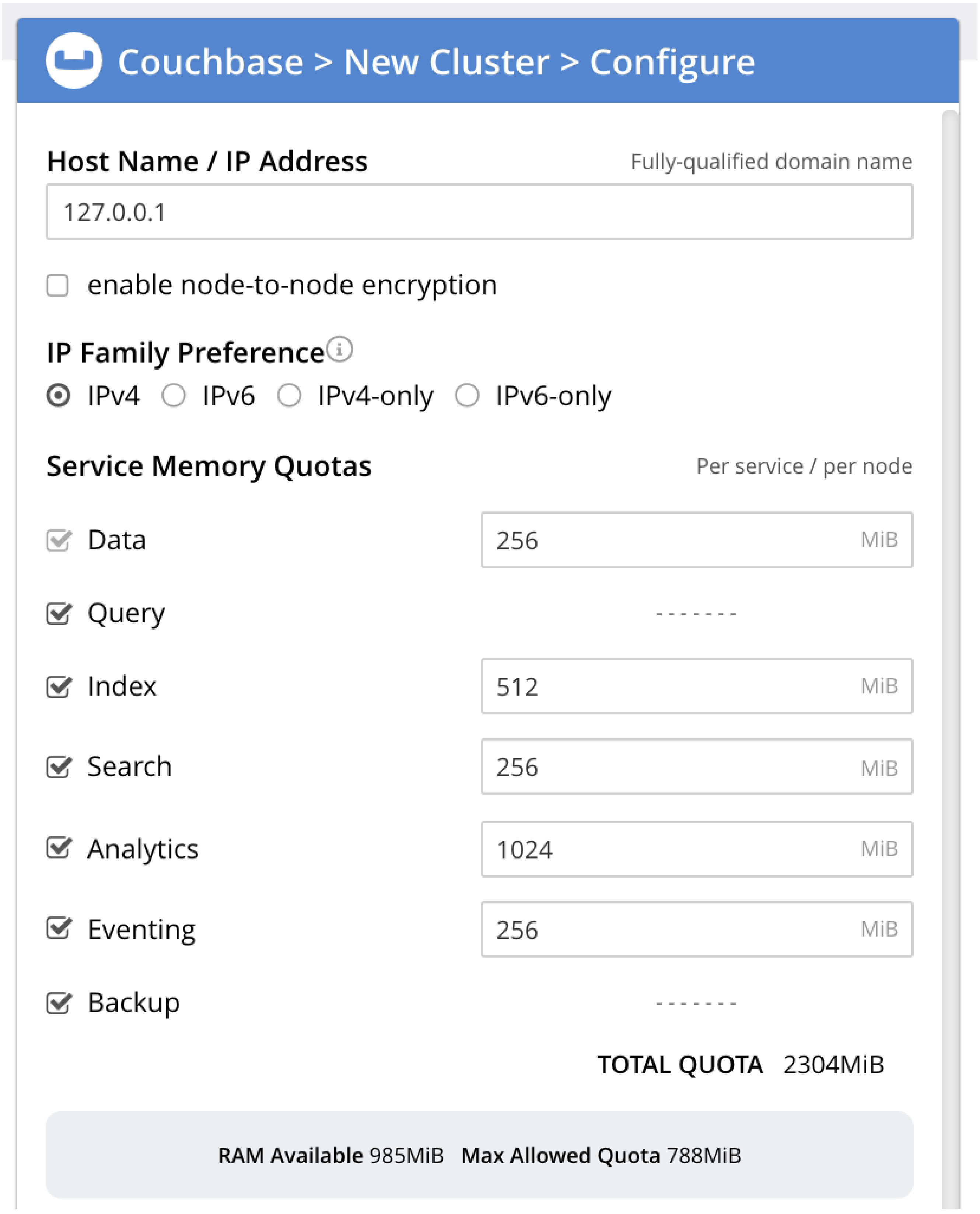
Privacy FAQ   Couchbase Privacy Policy

☑ Share usage information and get software update notifications.

< Back   Finish With Defaults   Configure Disk, Memory, Services

Picture 2-3

- Configure CouchBase Server



Couchbase > New Cluster > Configure

Host Name / IP Address — Fully-qualified domain name

127.0.0.1

☐ enable node-to-node encryption

IP Family Preference

[✓ IPv4] IPv6 IPv4-only IPv6-only

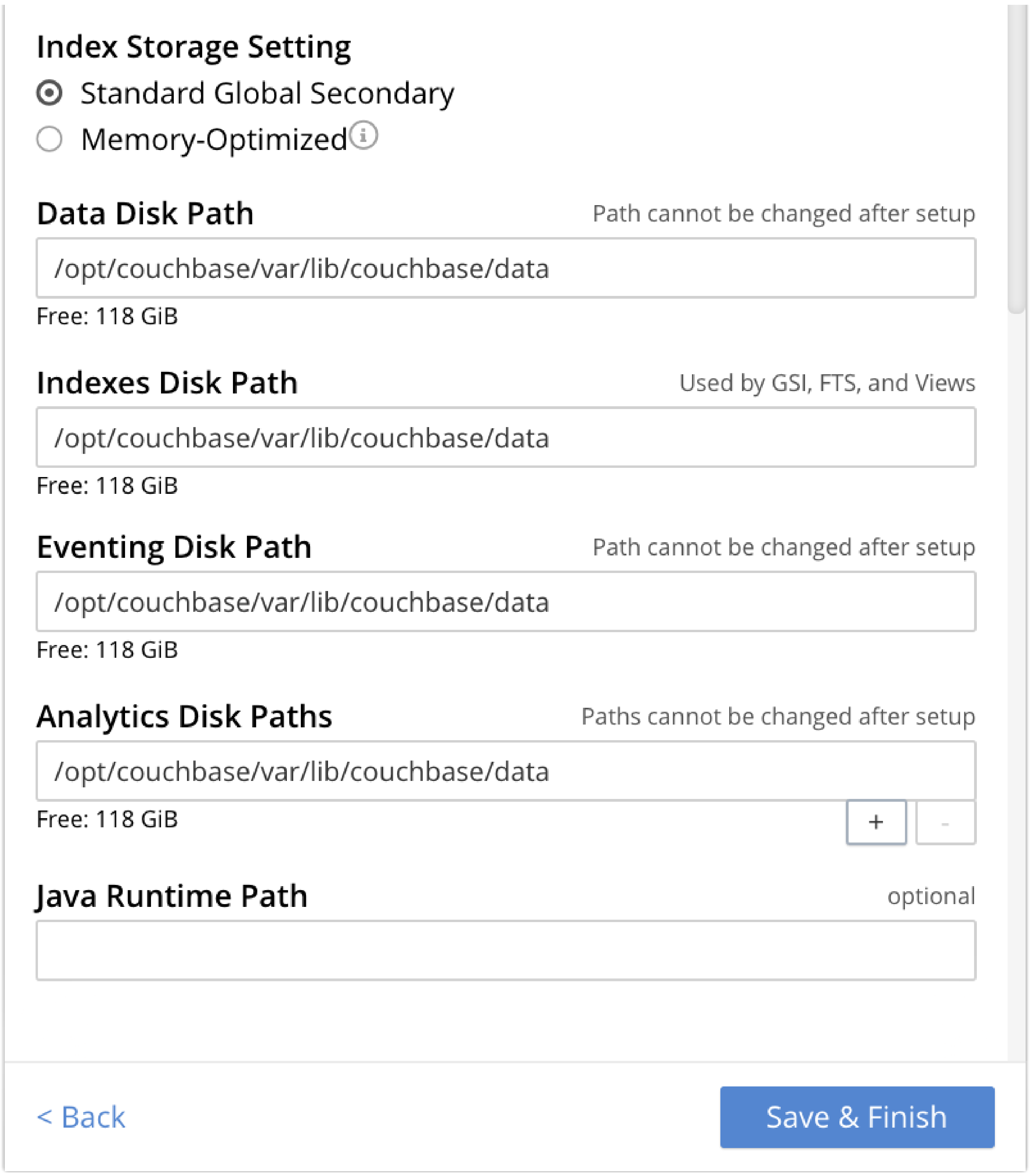
Service Memory Quotas — Per service / per node

✓ Data	256 MiB
✓ Query	-------
✓ Index	512 MiB
✓ Search	256 MiB
✓ Analytics	1024 MiB
✓ Eventing	256 MiB
✓ Backup	-------

TOTAL QUOTA 2304MiB

RAM Available 985MiB Max Allowed Quota 788MiB

Picture 2-4



Index Storage Setting

[✓ Standard Global Secondary] Memory-Optimized

Data Disk Path — Path cannot be changed after setup

/opt/couchbase/var/lib/couchbase/data

Free: 118 GiB

Indexes Disk Path — Used by GSI, FTS, and Views

/opt/couchbase/var/lib/couchbase/data

Free: 118 GiB

Eventing Disk Path — Path cannot be changed after setup

/opt/couchbase/var/lib/couchbase/data

Free: 118 GiB

Analytics Disk Paths — Paths cannot be changed after setup

/opt/couchbase/var/lib/couchbase/data

Free: 118 GiB

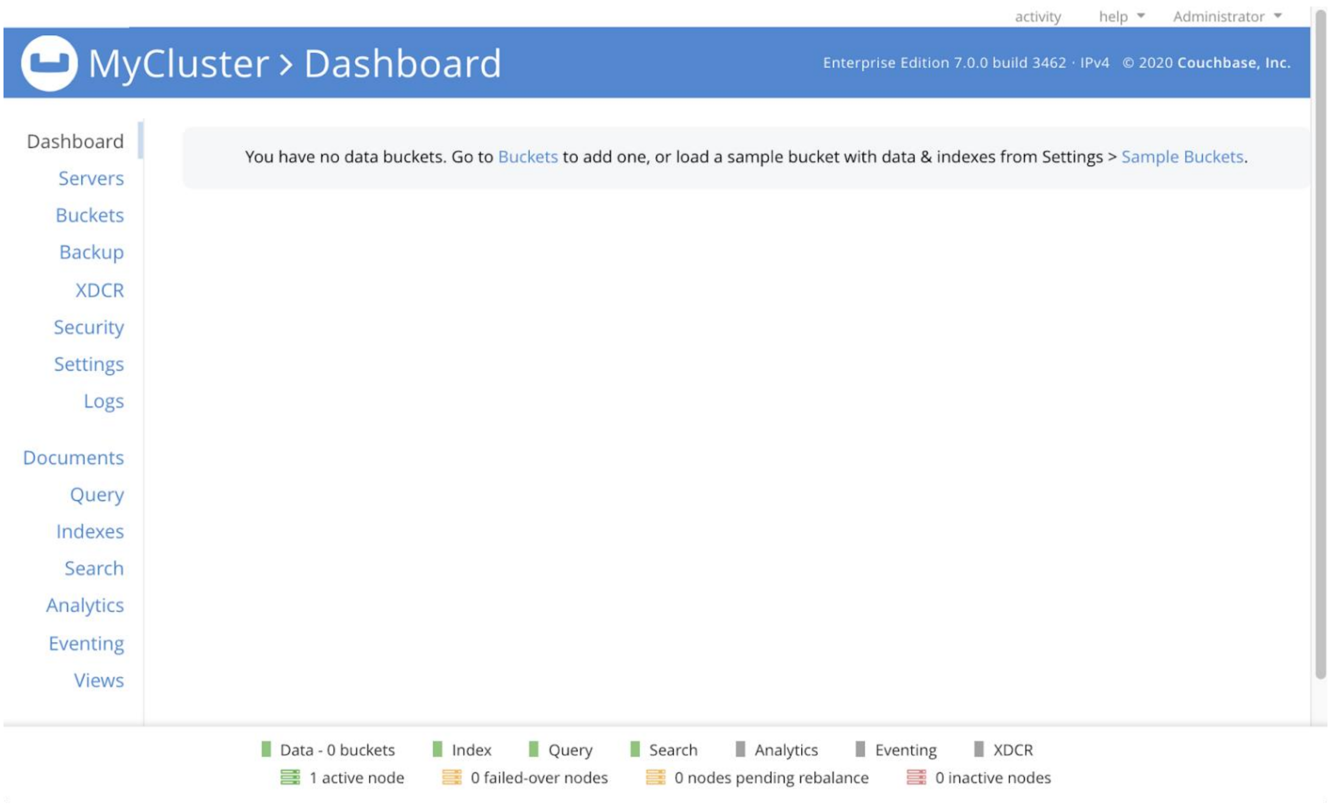
Java Runtime Path — optional

Picture 2-5

- Host Name/ IP Address: The name will be used for this node. You can use loopback address 127.0.0.1, when a second node is added to the cluster, the name will be changed to the IP of the underlying host. If you wish, you can substitute the IP address of the underlying host now, or you can substitute the fully qualified hostname of the underlying host

- enable node ecryption: Check the checkbox to enable node-to-node encryption for the cluster
- IP Family Preferences: Select IP V4(default)
- Service Memory Quotas: Allows you to specify how much memory should be allocated to each service you select for both current node and node subsequently add to the cluster
    - Data Service: 256Mib or more
    - Index Service: Selection and RAM-allocation to support Global Secondary Indexes. This should be 256 MiB or more
    - Query Service: No RAM-allocation
    - Search Service: Selection and RAM-allocation for the Full Text Search Service. This should be 256 MiB or more.
- Data and Index Disk Path: default location

- Then Click Save & Finish

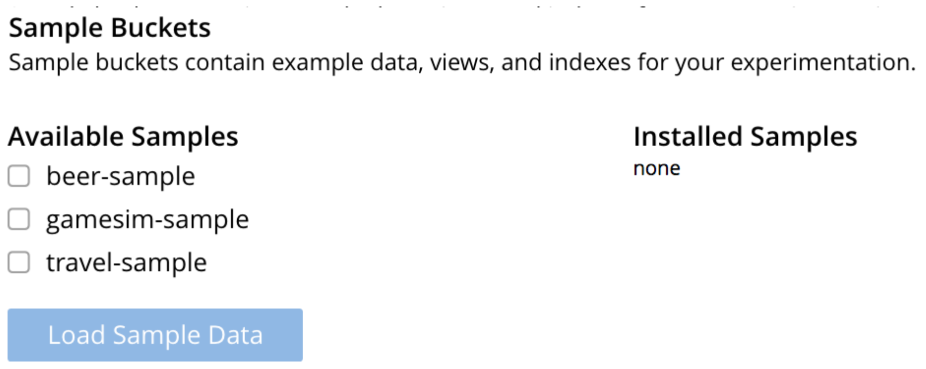


You have no data buckets. Go to Buckets to add one, or load a sample bucket with data & indexes from Settings > Sample Buckets.

Picture 2-6

2.1.2. Add Sample Bucket

- From the Settings screen, select the Sample Buckets tab. The Sample Buckets screen now appears, as follows:


Sample Buckets
Sample buckets contain example data, views, and indexes for your experimentation.

Available Samples
- ☐ beer-sample
- ☐ gamesim-sample
- ☐ travel-sample

Installed Samples
none

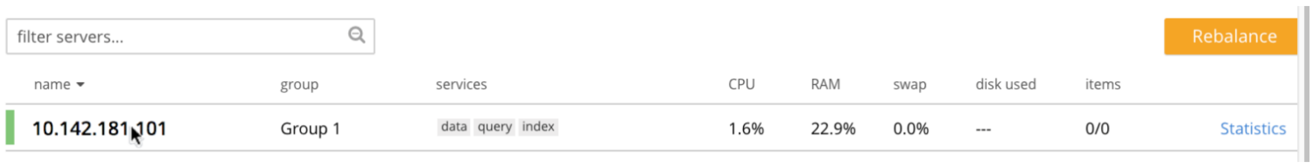
Picture 2-7

- Choose travel-sample and click Load Sample Data

2.2. Installation on cluster of nodes

2.2.1. Add A Node and Rebalance

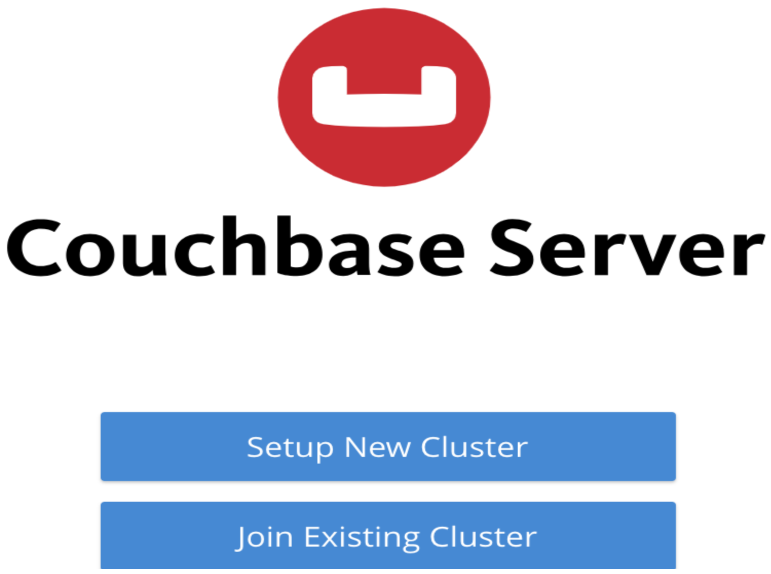
1. Access Server Screen



name	group	services	CPU	RAM	swap	disk used	items	
10.142.181.101	Group 1	data query index	1.6%	22.9%	0.0%	---	0/0	Statistics

Picture 2-8

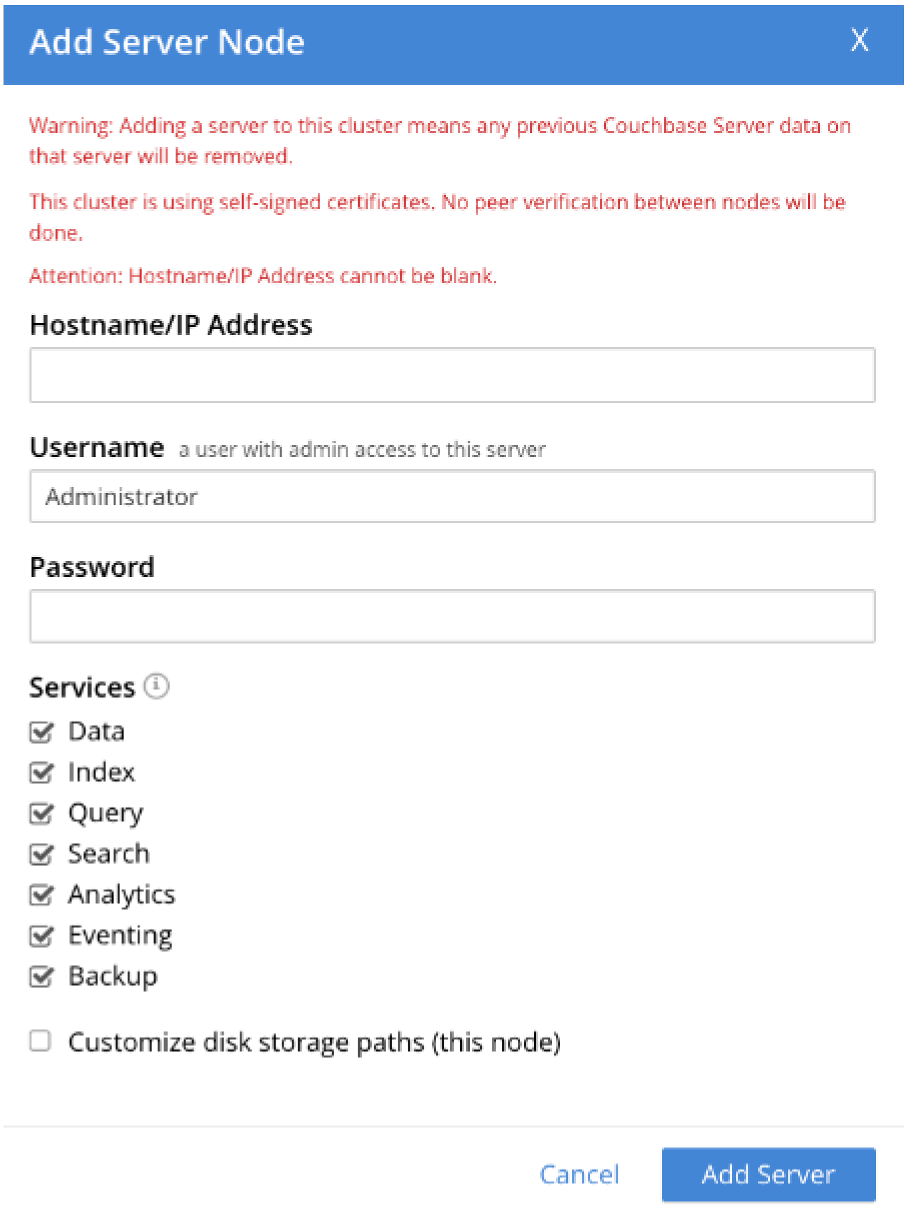
2. Ensure that the node to be added has been started. This can be accomplished by checking the IP address and port number for the new node in the address bar of the browser. The following interface is displayed:



Picture 2-9

This indicates that Couchbase Server is installed and running on the new node, but has not yet been provisioned. Do not use this interface: instead, return to Couchbase Web Console for the cluster, **10.142.181.101.**

3. Click ADD SERVER button



Add Server Node

Warning: Adding a server to this cluster means any previous Couchbase Server data on that server will be removed.

This cluster is using self-signed certificates. No peer verification between nodes will be done.

Attention: Hostname/IP Address cannot be blank.

Hostname/IP Address

Username a user with admin access to this server

Administrator

Password

Services

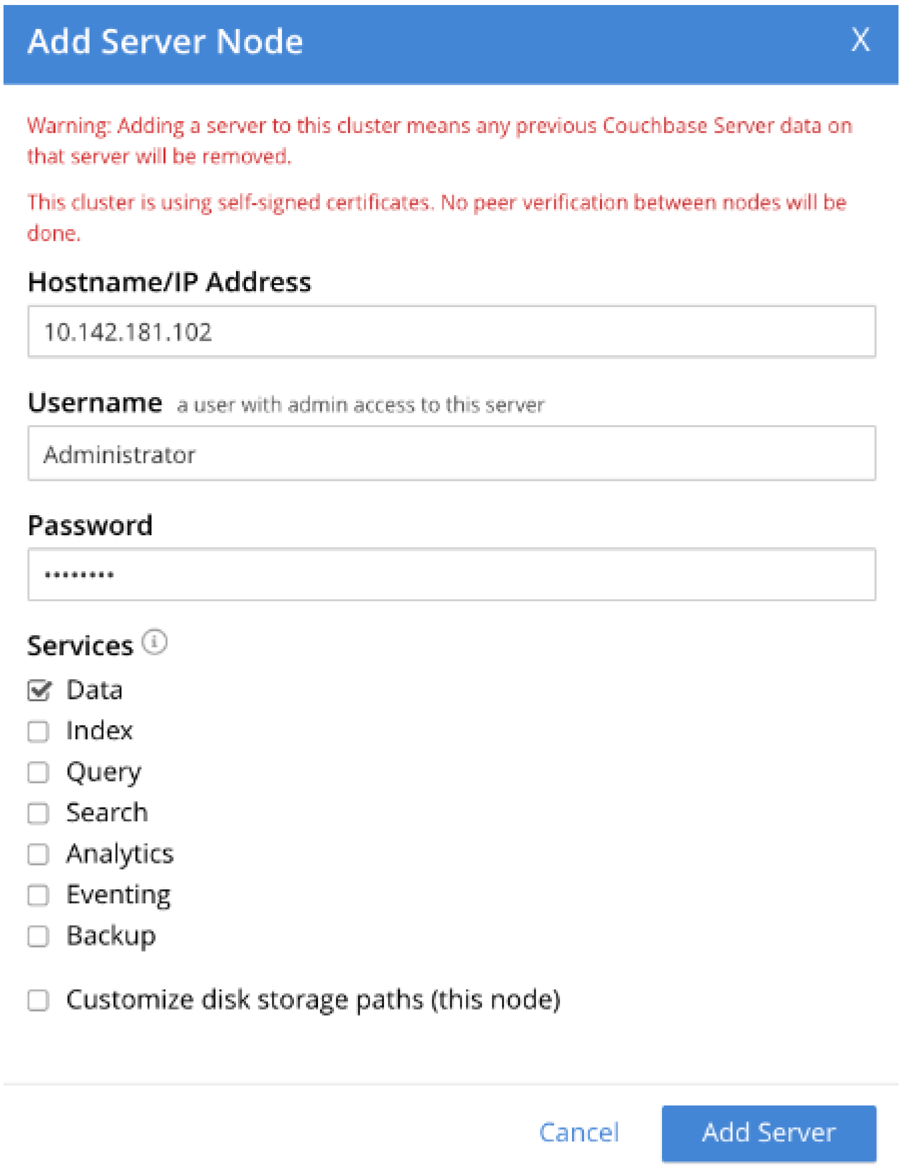
- [x] Data
- [x] Index
- [x] Query
- [x] Search
- [x] Analytics
- [x] Eventing
- [x] Backup

- [ ] Customize disk storage paths (this node)

Picture 2-10

Note the warning provided at the top of the dialog: if the node to be added has already been provisioned, the results of such provisioning will be eliminated and replaced on the node’s addition to the current cluster. (In fact, the node to be added in this example, has neither been initialized nor provisioned.)

4. Specify the IP address of the node to be added. A placeholder password must be specified, even though the node has not yet been provisioned with one. Uncheck all of the Services check-boxes except Data. The dialog now appears as follows:



Warning: Adding a server to this cluster means any previous Couchbase Server data on that server will be removed.

This cluster is using self-signed certificates. No peer verification between nodes will be done.

Hostname/IP Address: 10.142.181.102

Username a user with admin access to this server: Administrator

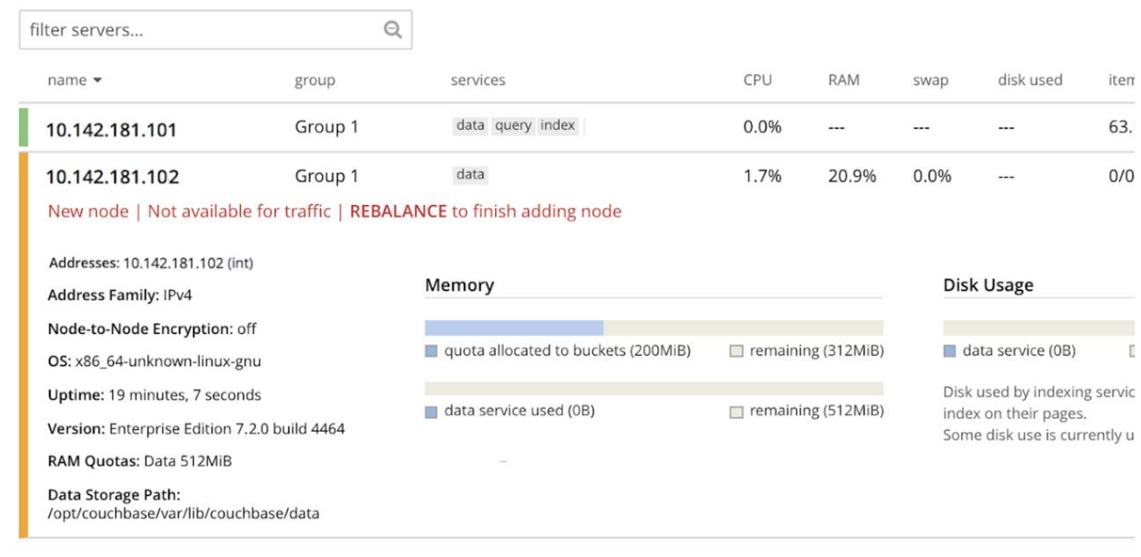
Password: ••••••••

Services
- [x] Data
- [ ] Index
- [ ] Query
- [ ] Search
- [ ] Analytics
- [ ] Eventing
- [ ] Backup

- [ ] Customize disk storage paths (this node)

Picture 2-11

Left-click on the Add Server button to save the settings. The Servers screen is redisplayed, with the following appearance:



filter servers...

name	group	services	CPU	RAM	swap	disk used	iten
10.142.181.101	Group 1	data query index	0.0%	---	---	---	63.
10.142.181.102	Group 1	data	1.7%	20.9%	0.0%	---	0/0

New node | Not available for traffic | REBALANCE to finish adding node

Addresses: 10.142.181.102 (int)
Address Family: IPv4
Node-to-Node Encryption: off
OS: x86_64-unknown-linux-gnu
Uptime: 19 minutes, 7 seconds
Version: Enterprise Edition 7.2.0 build 4464
RAM Quotas: Data 512MiB
Data Storage Path: /opt/couchbase/var/lib/couchbase/data

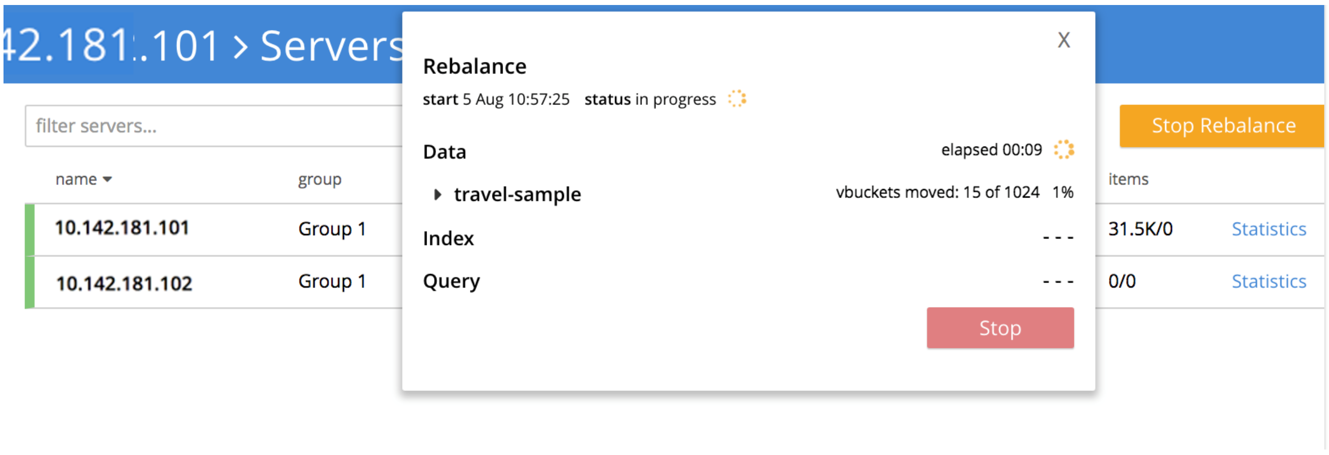
Memory
▨ quota allocated to buckets (200MiB) ☐ remaining (312MiB)
▨ data service used (0B) ☐ remaining (512MiB)

Disk Usage
▨ data service (0B)
Disk used by indexing servic
index on their pages.
Some disk use is currently u

This indicates that the new node, 10.142.181.102 has been successfully added. However, it is not yet taking traffic, and will be added following a *rebalance*. Note, at this point, the figure under the Items column for for 10.142.181.101: this is 63.1 K/0, which indicates that the node contains 63.1 K items in *active* vBuckets, and 0 items in *replica* vBuckets. Meanwhile, the Items figure for 10.142.181.102 is 0/0, indicating that no items are yet distributed onto that node in either active or replica form.

5. To perform a rebalance, left-click on the Rebalance button, at the upper right

The new node is rebalanced into the cluster, meaning that whatever active and replica vBuckets were previously distributed across the original cluster nodes are redistributed across the superset of nodes created by the addition. Additionally, a Rebalance dialog is displayed:

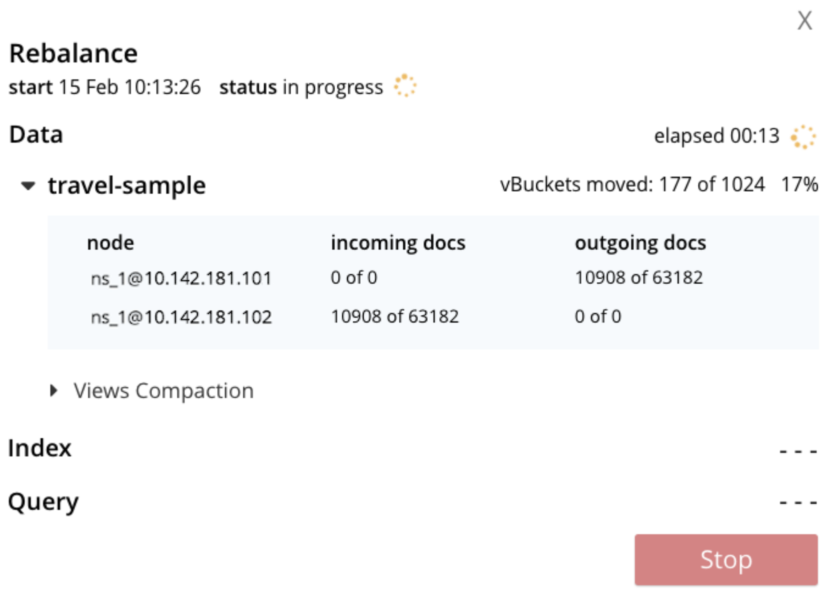


name	group	items
10.142.181.101	Group 1	31.5K/0
10.142.181.102	Group 1	0/0

Rebalance
start 5 Aug 10:57:25 status in progress
Data elapsed 00:09
travel-sample vbuckets moved: 15 of 1024 1%
Index - - -
Query - - -

Picture 2-12

The dialog indicates rebalance progress for each of the services on the cluster. To see more information on the progress related to the Data Service, left-click on the **travel-sample** tab, under **Data**: The pane expands to provide additional information on the progress of data-transfer for the **travel-sample** bucket:



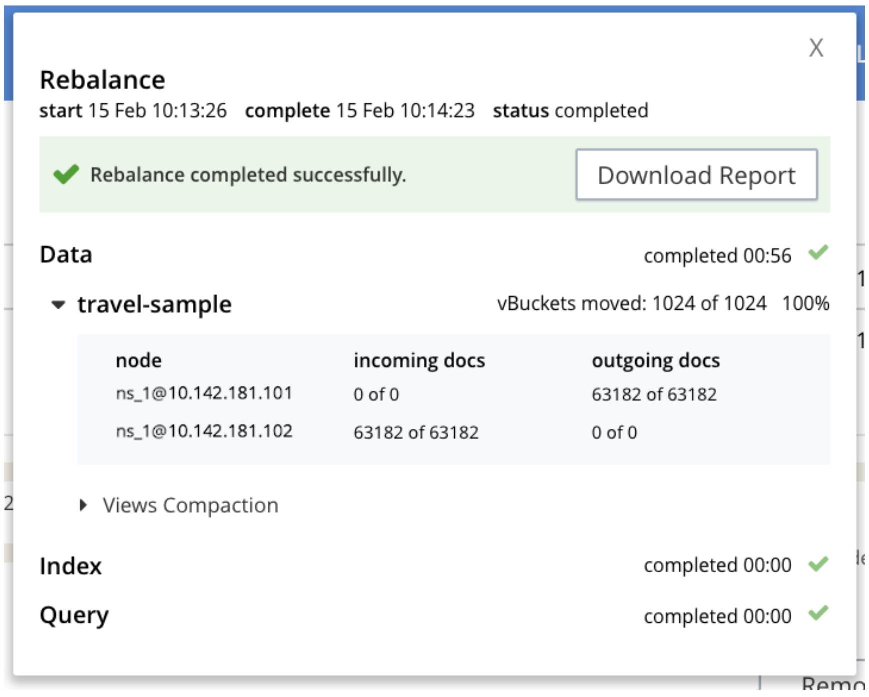
Rebalance
start 15 Feb 10:13:26 status in progress
Data elapsed 00:13
travel-sample vBuckets moved: 177 of 1024 17%

node	incoming docs	outgoing docs
ns_1@10.142.181.101	0 of 0	10908 of 63182
ns_1@10.142.181.102	10908 of 63182	0 of 0

Index - - -
Query - - -

Picture 2-13

When the rebalance is complete, the panel at the bottom of the screen disappears, and the dialog appears as follows:



X

Rebalance

start 15 Feb 10:13:26 **complete** 15 Feb 10:14:23 **status** completed

✓ Rebalance completed successfully.

Download Report

Data completed 00:56 ✓

▾ **travel-sample** vBuckets moved: 1024 of 1024 100%

node	incoming docs	outgoing docs
ns_1@10.142.181.101	0 of 0	63182 of 63182
ns_1@10.142.181.102	63182 of 63182	0 of 0

▸ Views Compaction

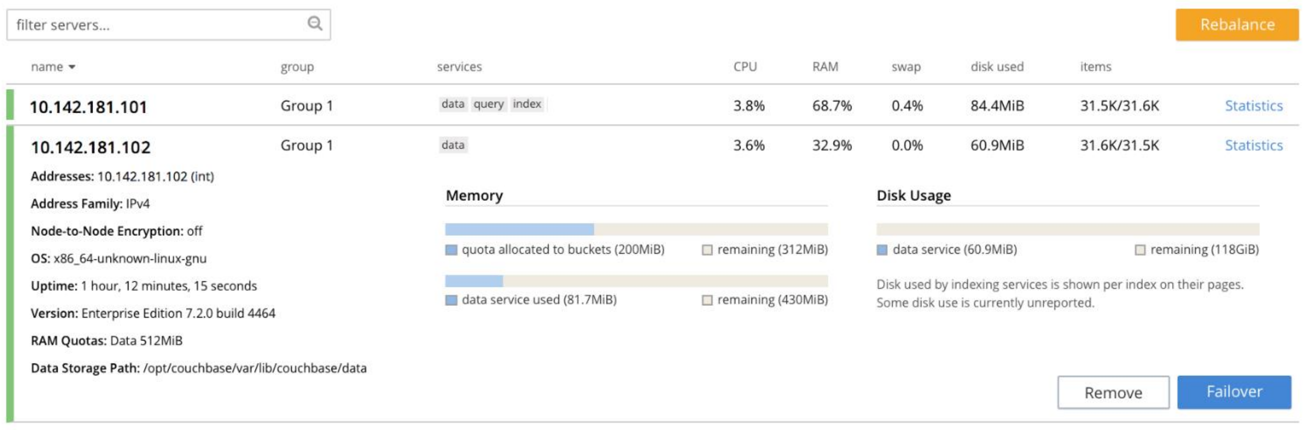
Index completed 00:00 ✓

Query completed 00:00 ✓

Picture 2-14

Left-click on the **X** at the upper-right of the dialog, to dismiss the dialog.

The **Servers** screen now displays two rows — one each for servers 10.142.181.101 and 10.142.181.102. Left-click on the row for 10.142.181.102, to open it. The screen now appears as follows:



filter servers...

Rebalance

name ▾	group	services	CPU	RAM	swap	disk used	items	
10.142.181.101	Group 1	data query index	3.8%	68.7%	0.4%	84.4MiB	31.5K/31.6K	Statistics
10.142.181.102	Group 1	data	3.6%	32.9%	0.0%	60.9MiB	31.6K/31.5K	Statistics

Addresses: 10.142.181.102 (int)
Address Family: IPv4
Node-to-Node Encryption: off
OS: x86_64-unknown-linux-gnu
Uptime: 1 hour, 12 minutes, 15 seconds
Version: Enterprise Edition 7.2.0 build 4464
RAM Quotas: Data 512MiB
Data Storage Path: /opt/couchbase/var/lib/couchbase/data

Memory

quota allocated to buckets (200MiB) remaining (312MiB)

data service used (81.7MiB) remaining (430MiB)

Disk Usage

data service (60.9MiB) remaining (118GiB)

Disk used by indexing services is shown per index on their pages.
Some disk use is currently unreported.

Remove Failover

Picture 2-15

Note that the figure in the **Items** column for node 10.142.181.101 is 31.5 K/31.6 K, which indicates that 31.5 K items are stored on the node in *active* vBuckets, and 31.6 K in *replica* vBuckets. The figure for 10.142.181.102 indicates the converse. Therefore, replication has successfully distributed the contents of travel-sample across both nodes, providing a single replica vBucket for each active vBucket.

3. Perform Distributed Stimulation

3.1. Description of problem posed

Simulating a distributed database system managing information about classes in an educational organization, used and accessed by multiple different applications. In each branch of the organization, there are applications that regularly access information about the classes conducted at that branch. Due to the geographical distance between branches, to optimize the overall system performance, classes will be distributed across nodes corresponding to the locations where they take place. The challenge is that there is a central application that needs to access information about all classes, even though the class information is distributed across multiple locations.

Details: Create 3 server nodes to store about class and subjects:

- Node A will store distributed data about class taught at branch A
- Node B will store distributed data about class taught at branch B
- Node C will access data from both A and B

3.2. Description of data structure used

Bucket: ClassesDetails

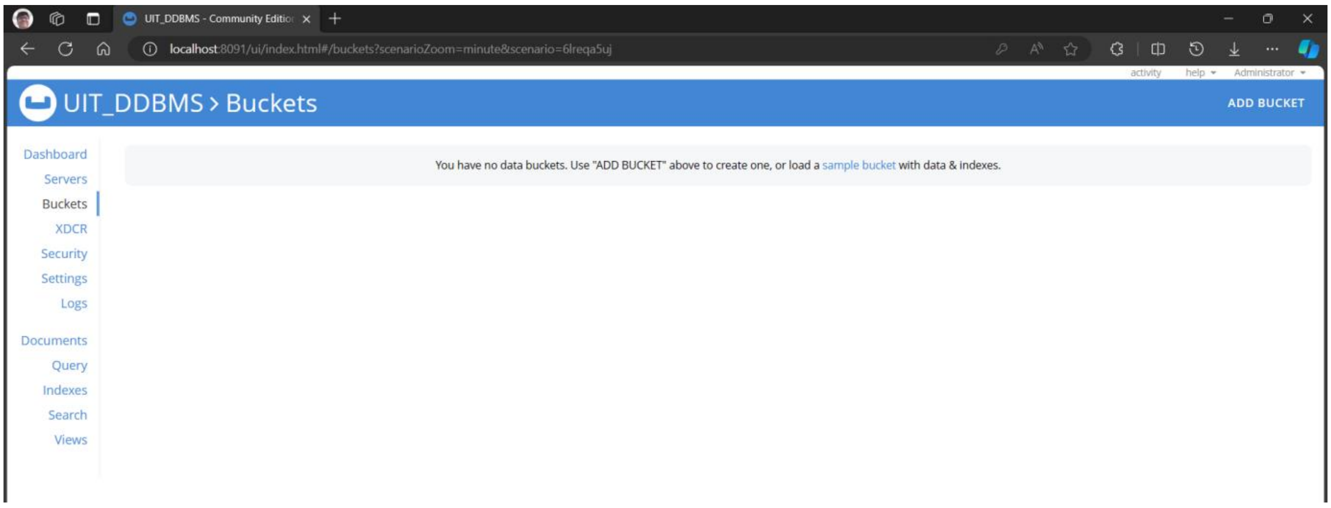
- classId: Class code.
- className: Class name.
- beginDate: Start date.
- finalDate: End date.
- location
- room
- site: Branch code, location where the class takes place.

3.3. Empirical steps

- Prepare:
    - Computer A (node A) – IP: 26.192.173.199

- Computer B (node B) – IP: 26.121.249.103
- Computer C (node C) – IP: 26.173.173.157
- Turn off fire wall in all node
- Node C will be an initial node and have a Bucket
- Node A and B will be a new node

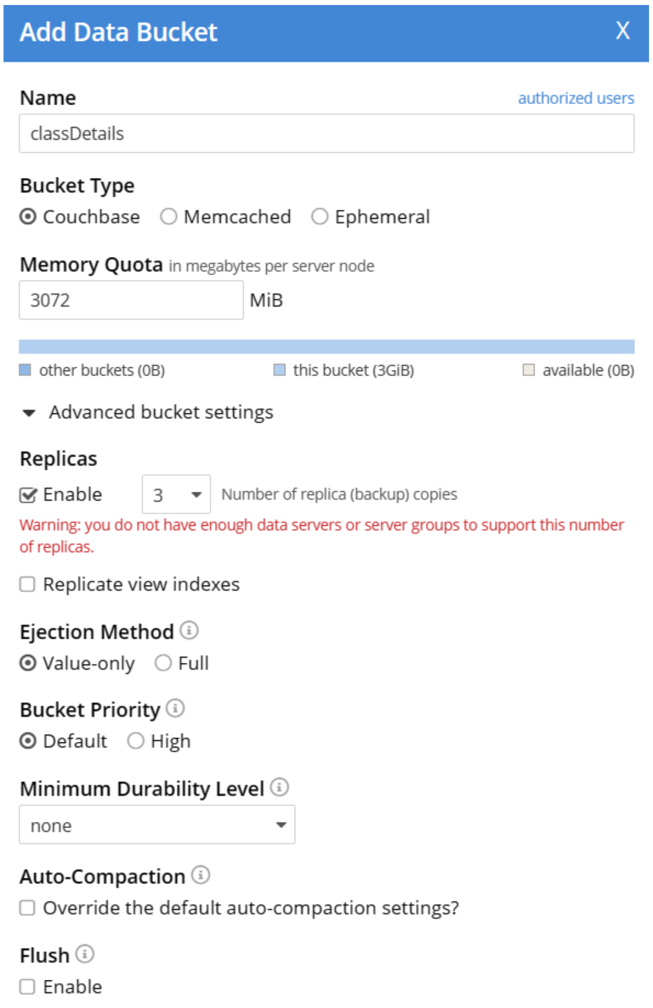
- **Config node C**
  - Add Bucket in node C:
- Choose Tab Buckets → Choose "ADD BUCKET" in the top right corner



You have no data buckets. Use "ADD BUCKET" above to create one, or load a sample bucket with data & indexes.

Picture 3-1

- A add form bucket will pop up → Fill out the information → Click "OK"



Name: classDetails

Bucket Type: [✓ Couchbase] Memcached Ephemeral

Memory Quota in megabytes per server node: 3072 MiB

🟦 other buckets (0B) 🟦 this bucket (3GiB) ⬜ available (0B)

Replicas
[✓ Enable] 3 Number of replica (backup) copies

Warning: you do not have enough data servers or server groups to support this number of replicas.

☐ Replicate view indexes

Ejection Method: [✓ Value-only] Full

Bucket Priority: [✓ Default] High

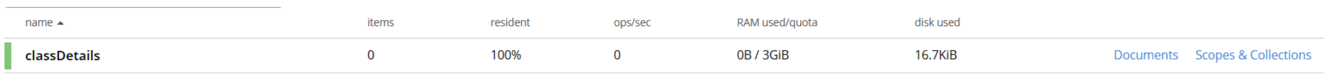
Minimum Durability Level: none

Auto-Compaction
☐ Override the default auto-compaction settings?

Flush
☐ Enable

Picture 3-2

- Bucket was added successfully



name	items	resident	ops/sec	RAM used/quota	disk used	
classDetails	0	100%	0	0B / 3GiB	16.7KiB	Documents Scopes & Collections

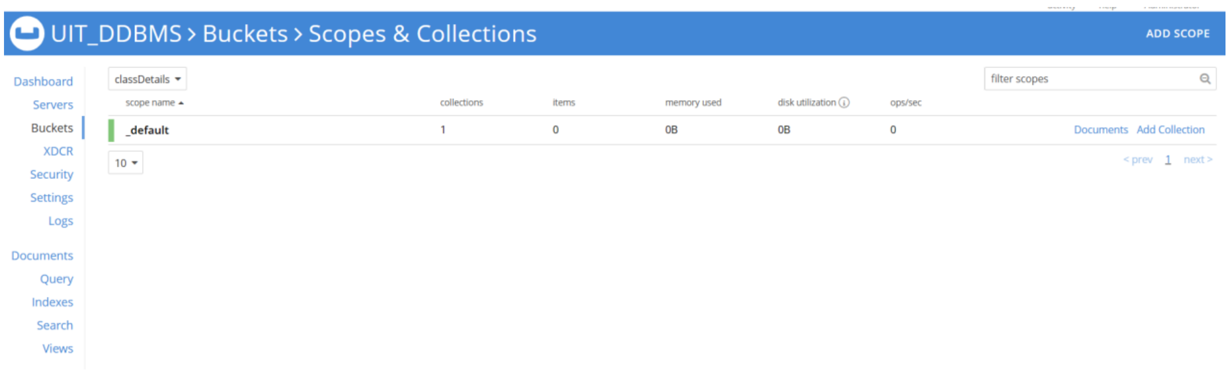
Picture 3-3

1.1. Add Scope and Collection:

- In the end of the Bucket item → Choose "Scopes & Collections"

Picture 3-4

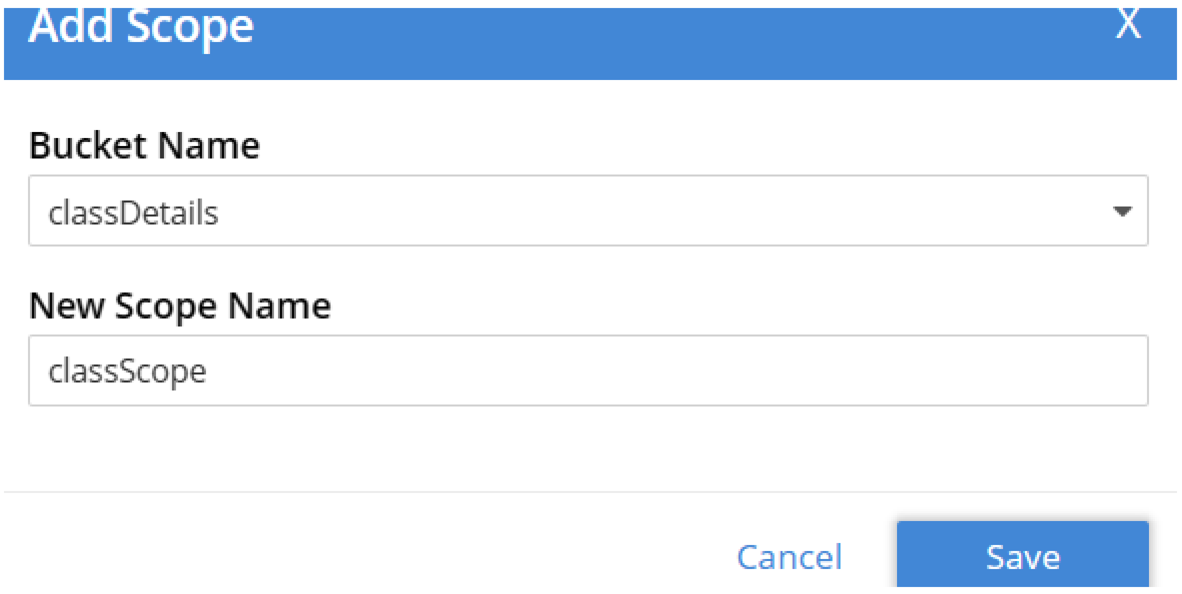
- List of Scopes will be shown → Choose "ADD SCOPE"



scope name	collections	items	memory used	disk utilization	ops/sec	
_default	1	0	0B	0B	0	Documents Add Collection

Picture 3-5

- Choose Bucket and fill out the name of scope we want to creata (Ex: classScope)



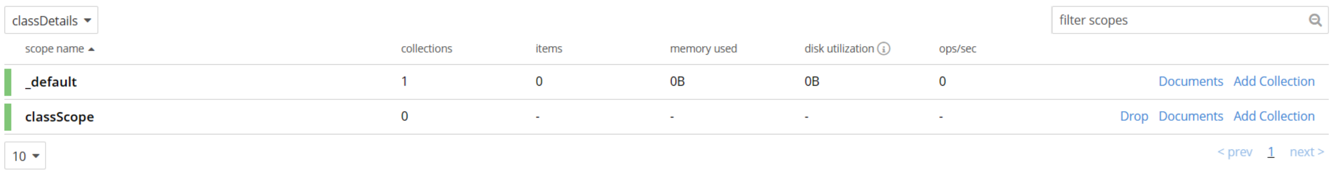
Add Scope

Bucket Name: classDetails

New Scope Name: classScope

Picture 3-6

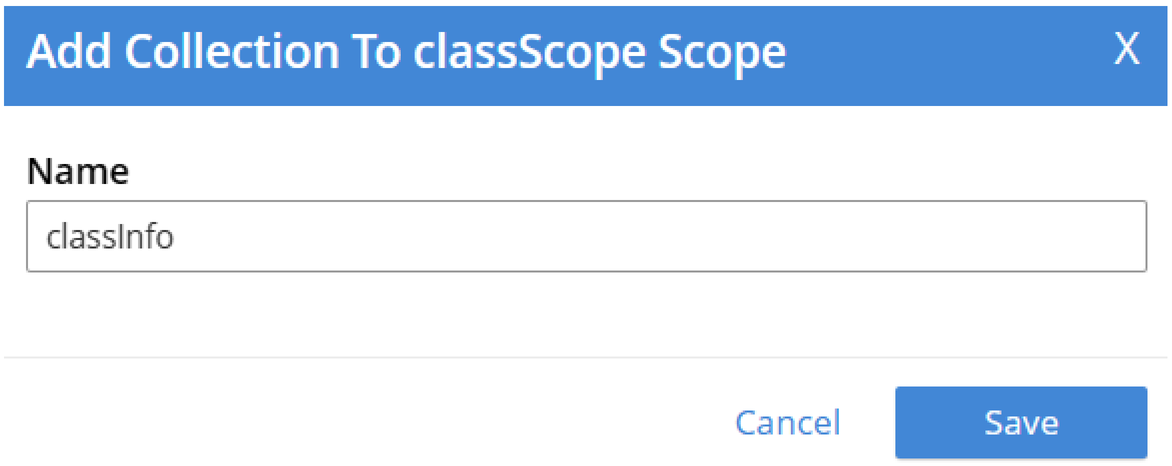
- Scope classScope was added successfully



scope name	collections	items	memory used	disk utilization	ops/sec	
_default	1	0	0B	0B	0	Documents Add Collection
classScope	0	-	-	-	-	Drop Documents Add Collection

Picture 3-7

- In the end of that Scope → Choose "Add Collection" → Fill out the name of Collection
  (EX: classInfo) → Save



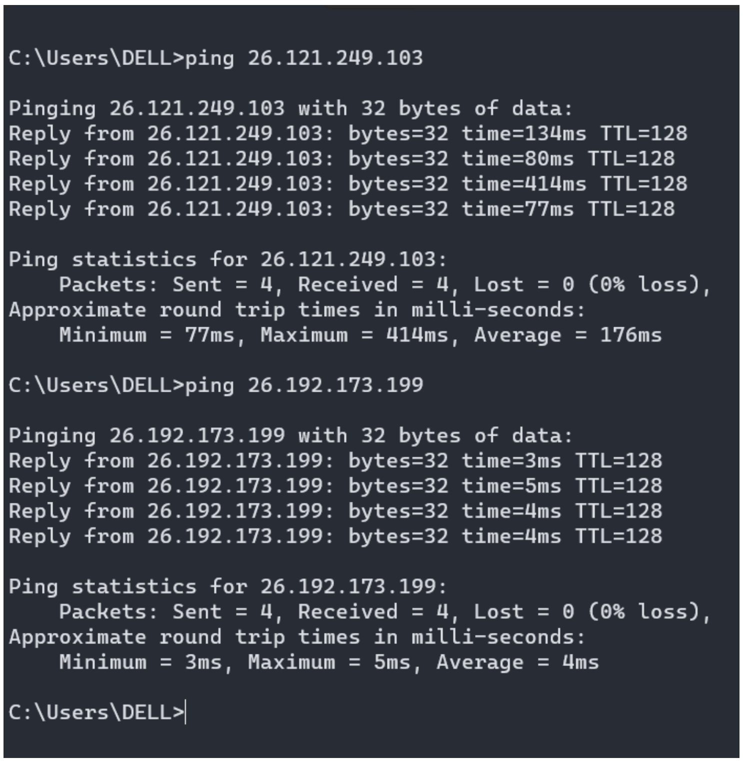
Add Collection To classScope Scope

Name: classInfo

Picture 3-8

2. Add node A and B:

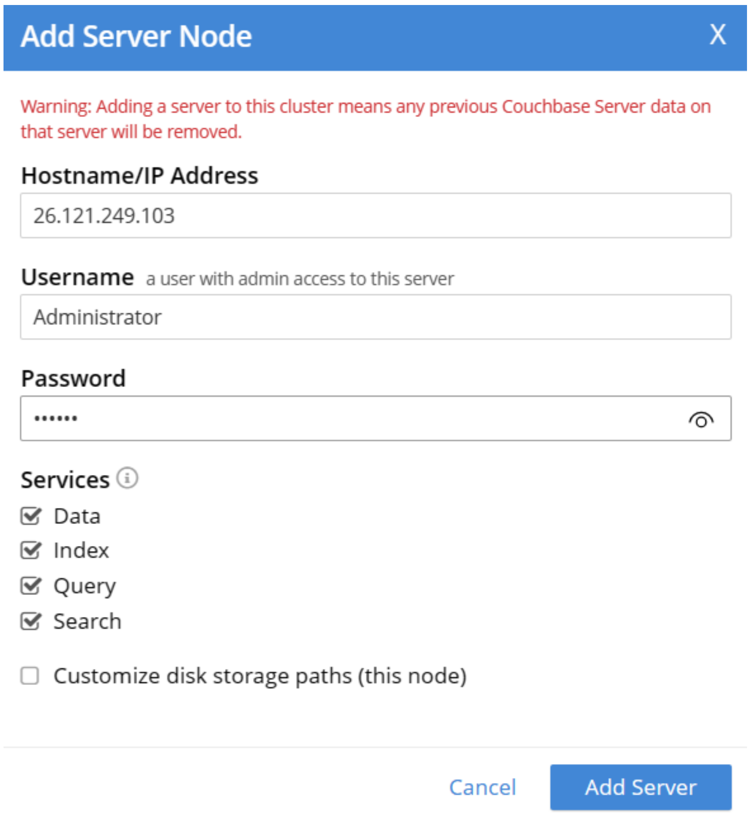
- Check the connection between node A and C, node B and C



```
C:\Users\DELL>ping 26.121.249.103

Pinging 26.121.249.103 with 32 bytes of data:
Reply from 26.121.249.103: bytes=32 time=134ms TTL=128
Reply from 26.121.249.103: bytes=32 time=80ms TTL=128
Reply from 26.121.249.103: bytes=32 time=414ms TTL=128
Reply from 26.121.249.103: bytes=32 time=77ms TTL=128

Ping statistics for 26.121.249.103:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 77ms, Maximum = 414ms, Average = 176ms

C:\Users\DELL>ping 26.192.173.199

Pinging 26.192.173.199 with 32 bytes of data:
Reply from 26.192.173.199: bytes=32 time=3ms TTL=128
Reply from 26.192.173.199: bytes=32 time=5ms TTL=128
Reply from 26.192.173.199: bytes=32 time=4ms TTL=128
Reply from 26.192.173.199: bytes=32 time=4ms TTL=128

Ping statistics for 26.192.173.199:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 3ms, Maximum = 5ms, Average = 4ms

C:\Users\DELL>
```

Picture 3-9

- Choose Tab Buckets → Choose “ADD SERVER” in the top right corner → Enter the IP of the node you want to connect (EX: 26.121.249.103 – node B) and account have admin access


Add Server Node

Warning: Adding a server to this cluster means any previous Couchbase Server data on that server will be removed.

Hostname/IP Address: 26.121.249.103

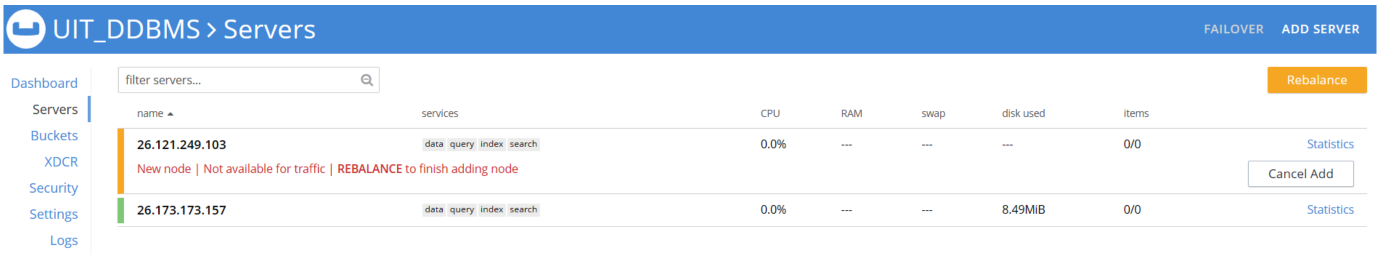
Username (a user with admin access to this server): Administrator

Password: ••••••

Services
- [✓] Data
- [✓] Index
- [✓] Query
- [✓] Search
- [ ] Customize disk storage paths (this node)

Picture 3-10

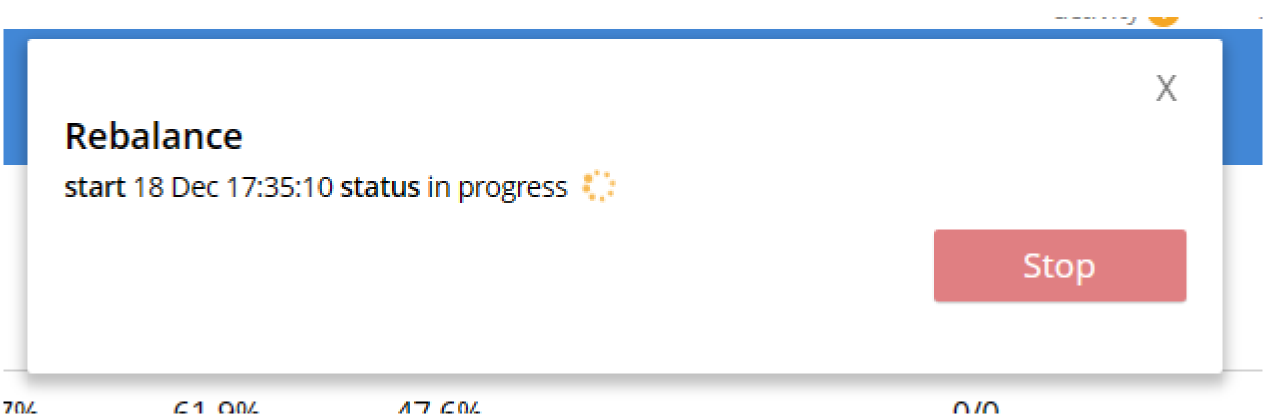
- Add node successfully → then start “Rebalance” → click “Rebalance” button



name	services	CPU	RAM	swap	disk used	items	
26.121.249.103 New node | Not available for traffic | **REBALANCE** to finish adding node	data query index search	0.0%	---	---	---	0/0	Statistics Cancel Add
26.173.173.157	data query index search	0.0%	---	---	8.49MiB	0/0	Statistics

Picture 3-11

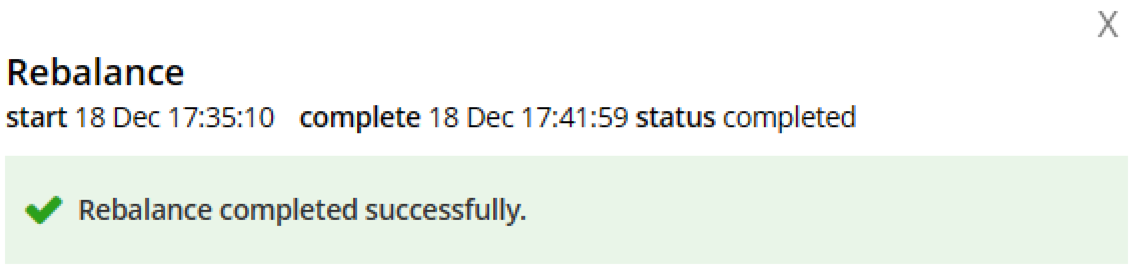
- Notificate that Rebalance is starting


Rebalance

start 18 Dec 17:35:10 status in progress

Picture 3-12

- Rebalance successfully



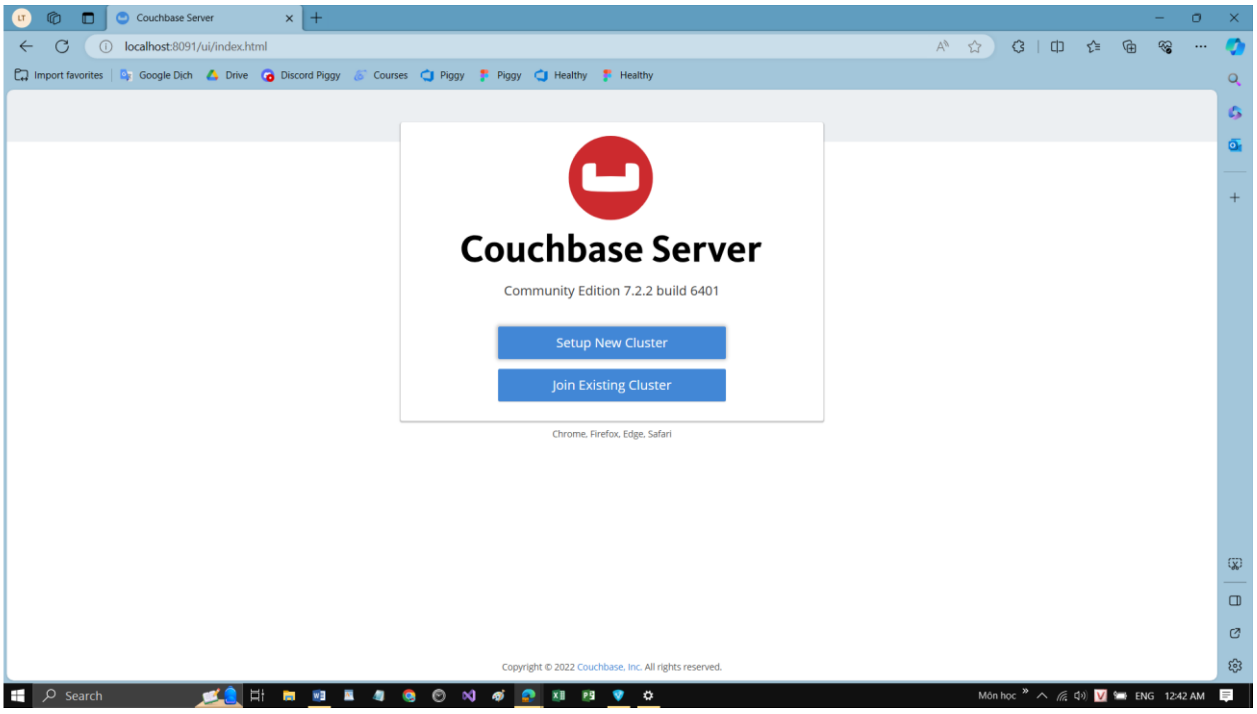
Rebalance

start 18 Dec 17:35:10 complete 18 Dec 17:41:59 status completed

Rebalance completed successfully.

Picture 3-13

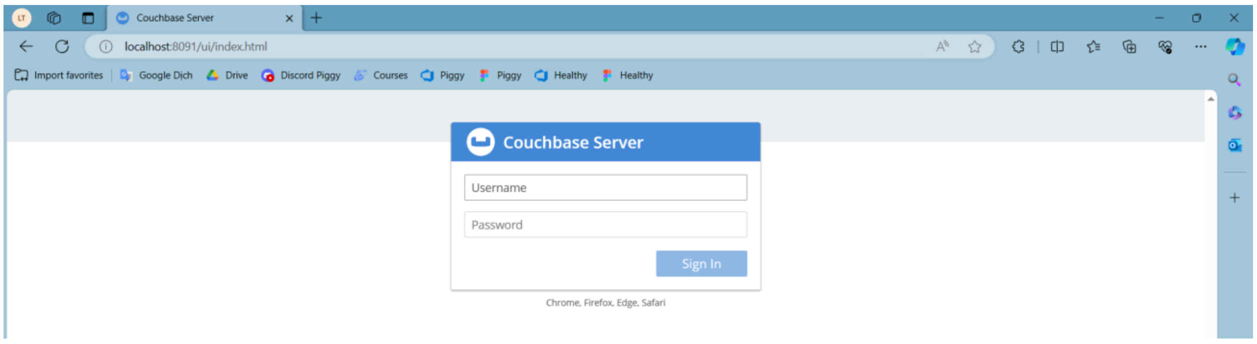
- Go to node B, this is first UI we have seen



Couchbase Server
localhost:8091/ui/index.html
Couchbase Server
Community Edition 7.2.2 build 6401
Setup New Cluster
Join Existing Cluster
Chrome, Firefox, Edge, Safari
Copyright © 2022 Couchbase, Inc. All rights reserved.

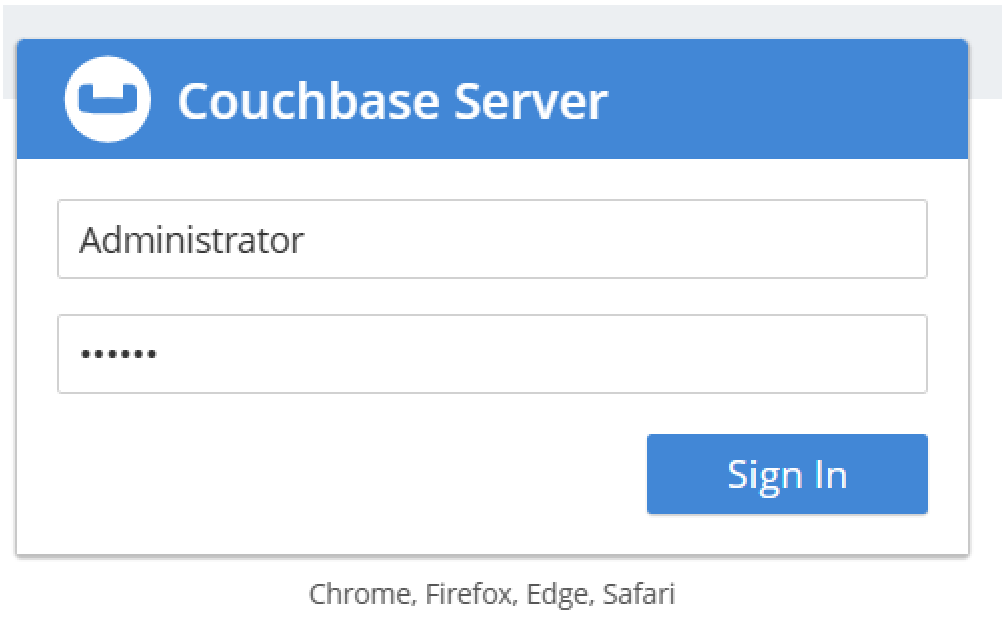
Picture 3-14

- After node B was added, refresh the page → Sign in UI will be displayed



Picture 3-15

- Use Admin account to sign in and check


Couchbase Server

Administrator

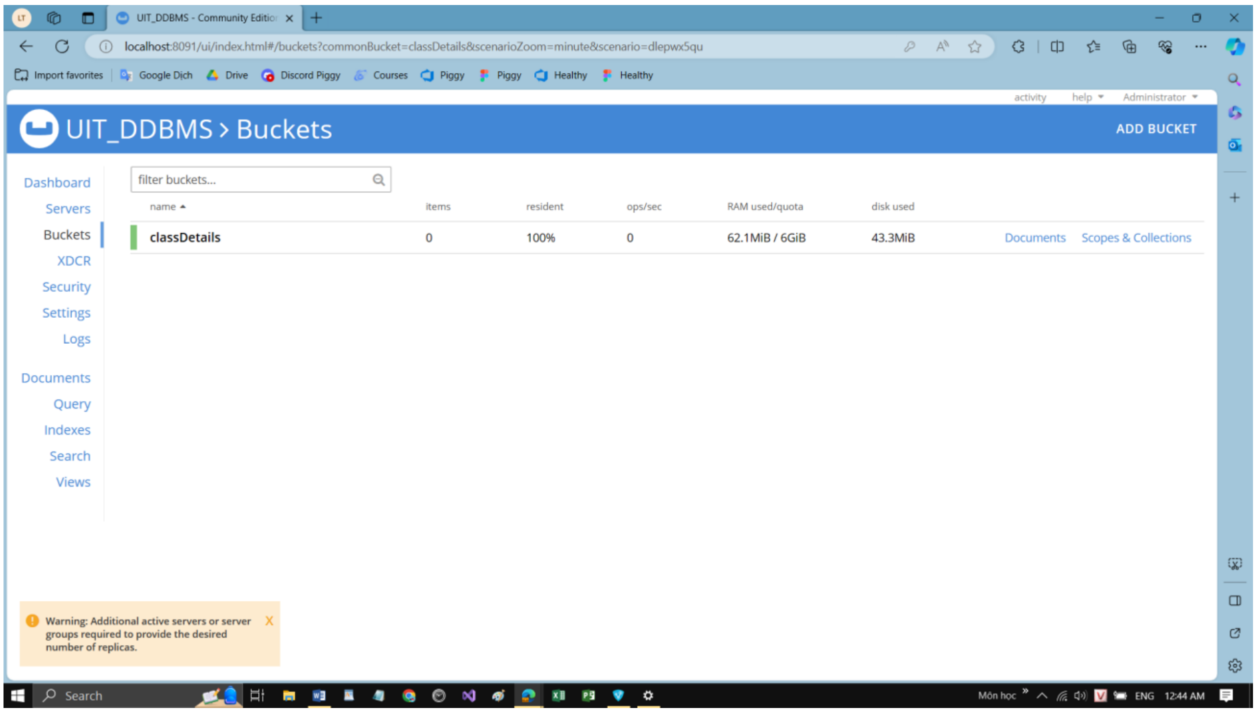
••••••

Sign In

Chrome, Firefox, Edge, Safari

Picture 3-16

- Bucket was synced from node C


localhost:8091/ui/index.html#/buckets?commonBucket=classDetails&scenarioZoom=minute&scenario=dlepwx5qu

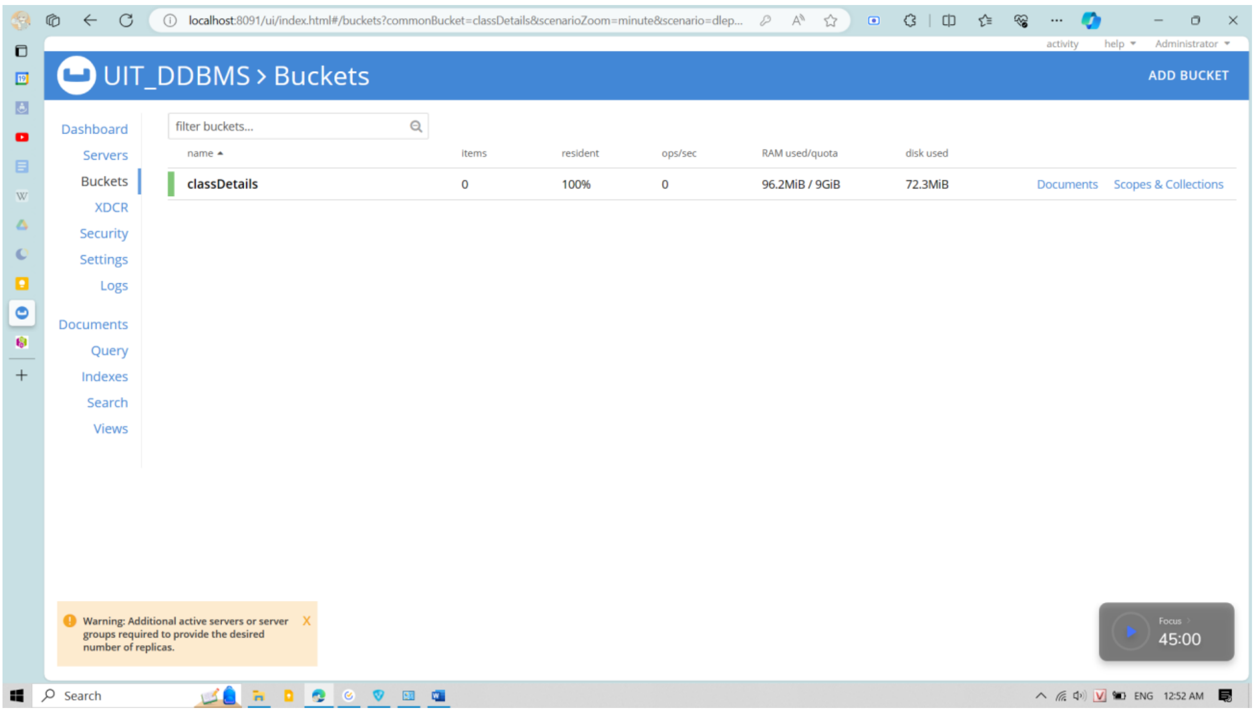
UIT_DDBMS > Buckets

name	items	resident	ops/sec	RAM used/quota	disk used		
classDetails	0	100%	0	62.1MiB / 6GiB	43.3MiB	Documents	Scopes & Collections

Warning: Additional active servers or server groups required to provide the desired number of replicas.

Picture 3-17

- And node A also

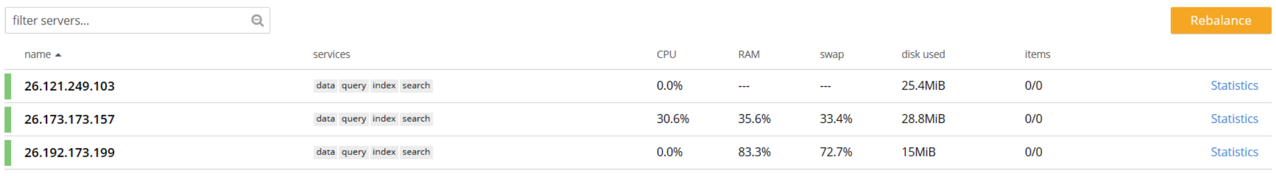


name	items	resident	ops/sec	RAM used/quota	disk used		
classDetails	0	100%	0	96.2MiB / 9GiB	72.3MiB	Documents	Scopes & Collections

Warning: Additional active servers or server groups required to provide the desired number of replicas.

Picture 3-18

- List of node in cluster



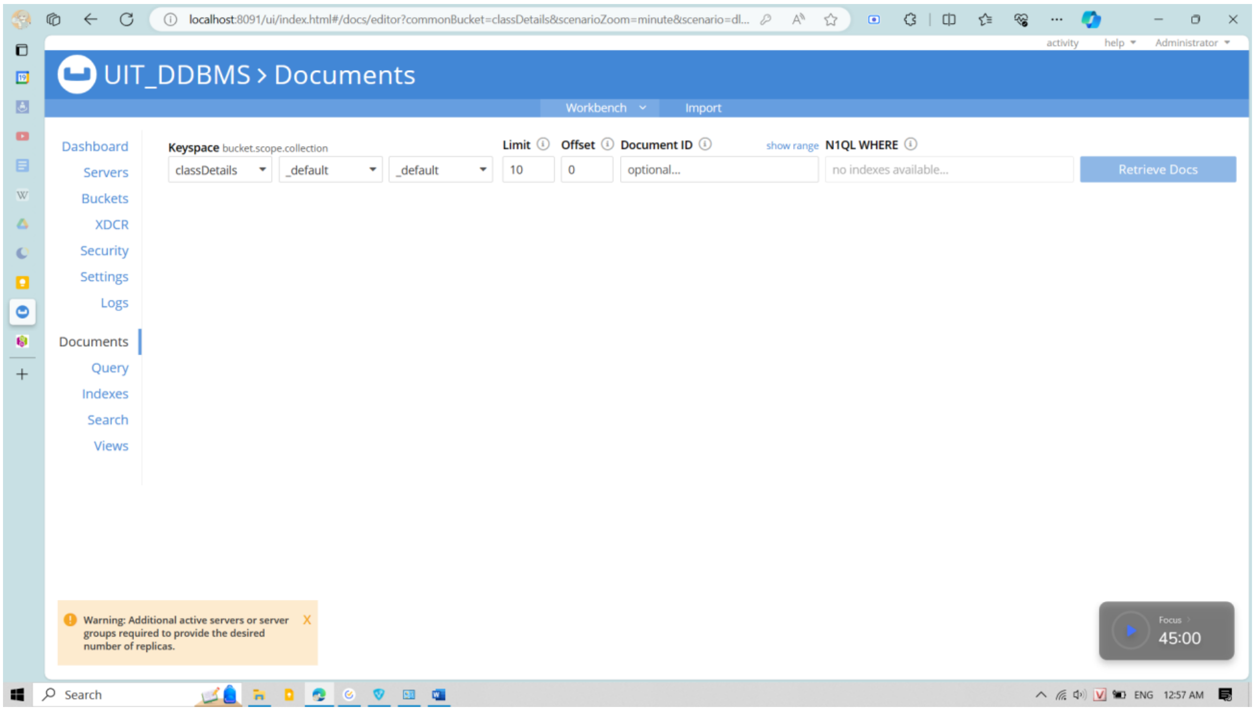
name	services	CPU	RAM	swap	disk used	items	
26.121.249.103	data query index search	0.0%	---	---	25.4MiB	0/0	Statistics
26.173.173.157	data query index search	30.6%	35.6%	33.4%	28.8MiB	0/0	Statistics
26.192.173.199	data query index search	0.0%	83.3%	72.7%	15MiB	0/0	Statistics

Picture 3-19

3. Import document and query:

3.1. Import document from node A and B:

- Choose tab "Documents"



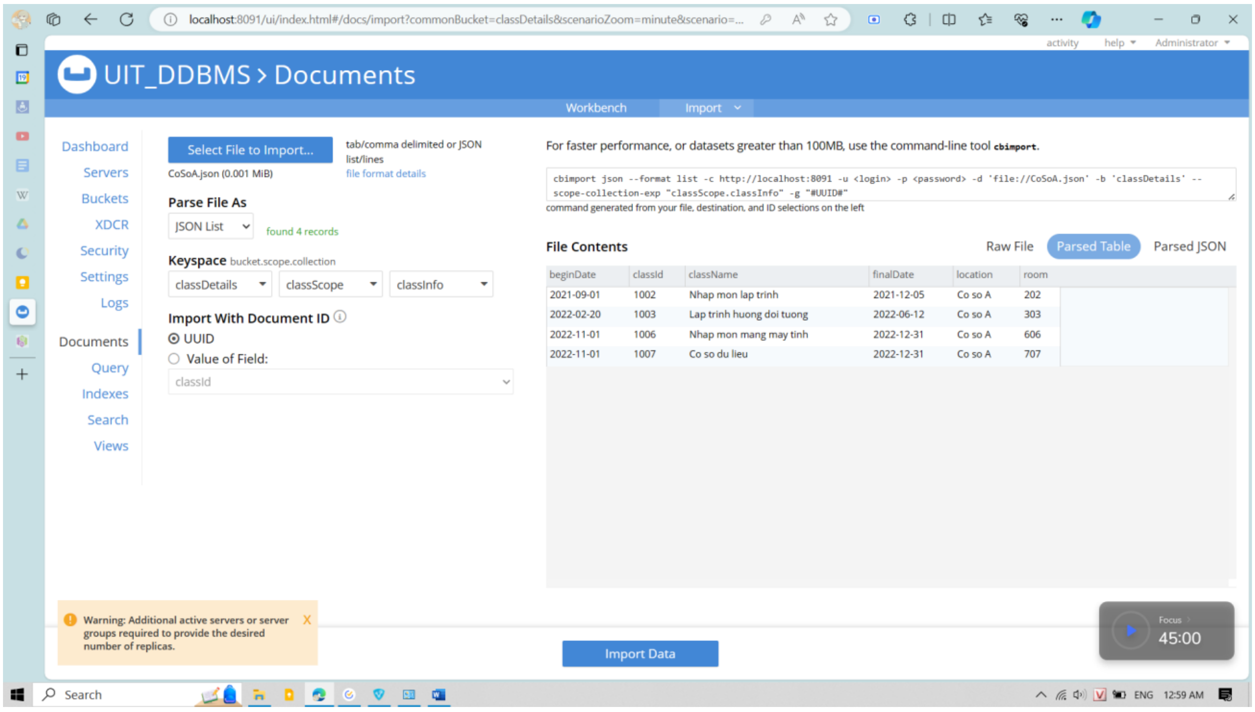
UIT_DDBMS > Documents

Keyspace bucket.scope.collection: classDetails, _default, _default — Limit: 10 — Offset: 0 — Document ID: optional... — N1QL WHERE: no indexes available...

Warning: Additional active servers or server groups required to provide the desired number of replicas.

Picture 3-20

- Choose "Import" in the middle top



UIT_DDBMS > Documents

Select File to Import... — tab/comma delimited or JSON list/lines — CoSoA.json (0.001 MiB)

Parse File As: JSON List — found 4 records

Keyspace bucket.scope.collection: classDetails, classScope, classInfo

Import With Document ID: [✓ UUID] Value of Field: classId

For faster performance, or datasets greater than 100MB, use the command-line tool `cbimport`.

```
cbimport json --format list -c http://localhost:8091 -u <login> -p <password> -d 'file://CoSoA.json' -b 'classDetails' --
scope-collection-exp "classScope.classInfo" -g "#UUID#"
```

command generated from your file, destination, and ID selections on the left

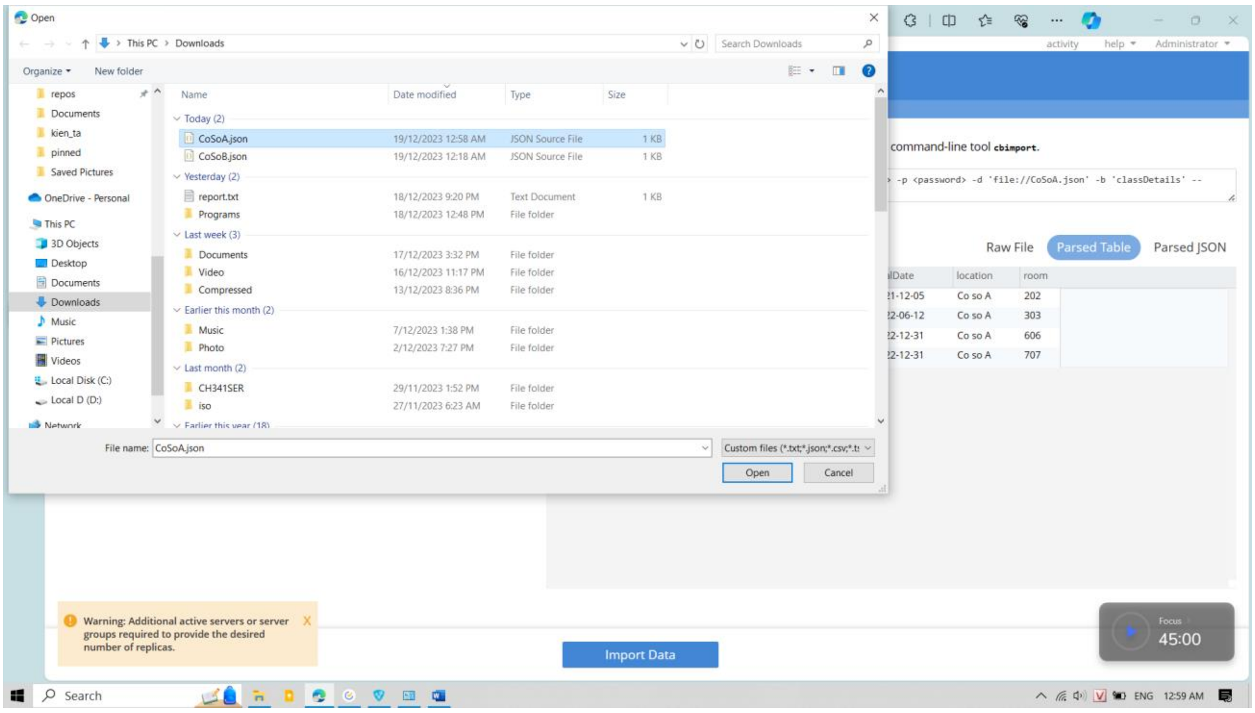
File Contents

beginDate	classId	className	finalDate	location	room
2021-09-01	1002	Nhap mon lap trinh	2021-12-05	Co so A	202
2022-02-20	1003	Lap trinh huong doi tuong	2022-06-12	Co so A	303
2022-11-01	1006	Nhap mon mang may tinh	2022-12-31	Co so A	606
2022-11-01	1007	Co so du lieu	2022-12-31	Co so A	707

Warning: Additional active servers or server groups required to provide the desired number of replicas.

Picture 3-21

- Choose “Select File to Import” button


Open

This PC › Downloads

Name	Date modified	Type	Size
Today (2)			
CoSoA.json	19/12/2023 12:58 AM	JSON Source File	1 KB
CoSoB.json	19/12/2023 12:18 AM	JSON Source File	1 KB
Yesterday (2)			
report.txt	18/12/2023 9:20 PM	Text Document	1 KB
Programs	18/12/2023 12:48 PM	File folder	
Last week (3)			
Documents	17/12/2023 3:32 PM	File folder	
Video	16/12/2023 11:17 PM	File folder	
Compressed	13/12/2023 8:36 PM	File folder	
Earlier this month (2)			
Music	7/12/2023 1:38 PM	File folder	
Photo	2/12/2023 7:27 PM	File folder	
Last month (2)			
CH341SER	29/11/2023 1:52 PM	File folder	
iso	27/11/2023 6:23 AM	File folder	
Earlier this year (18)			

File name: CoSoA.json

Custom files (*.txt;*.json;*.csv;*.t

Open Cancel

command-line tool **cbimport**.

```
-p <password> -d 'file://CoSoA.json' -b 'classDetails' --
```

Raw File | Parsed Table | Parsed JSON

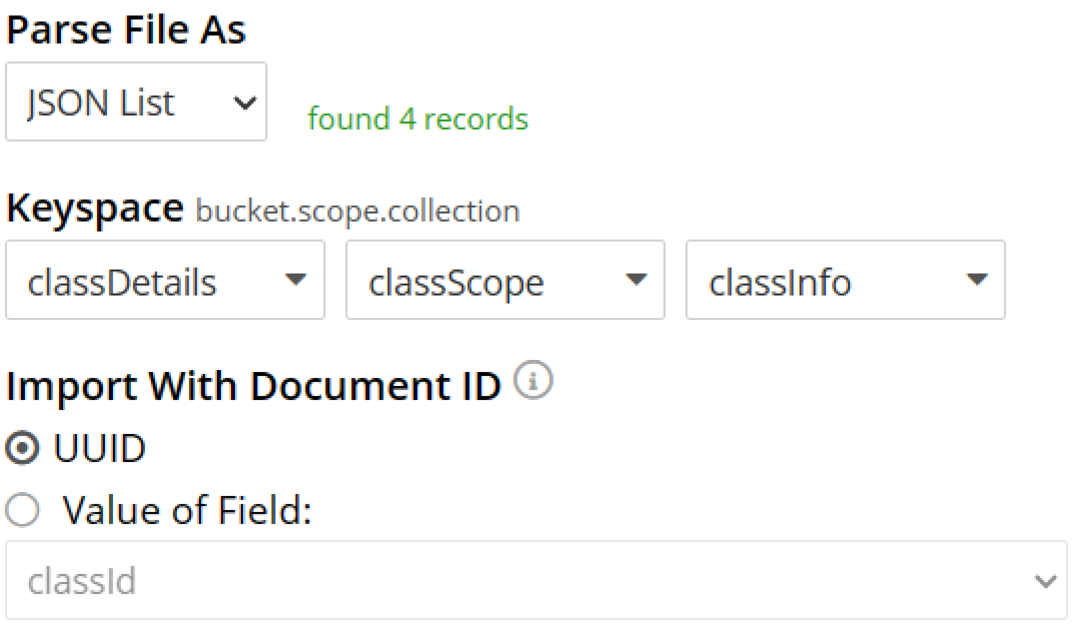
lDate	location	room
21-12-05	Co so A	202
22-06-12	Co so A	303
22-12-31	Co so A	606
22-12-31	Co so A	707

Warning: Additional active servers or server groups required to provide the desired number of replicas.

Import Data

Picture 3-22

- Choose file type and bucket.scope.collection → Then click “import data” button in the bottom middle


Parse File As

JSON List — found 4 records

Keyspace bucket.scope.collection

classDetails | classScope | classInfo

Import With Document ID

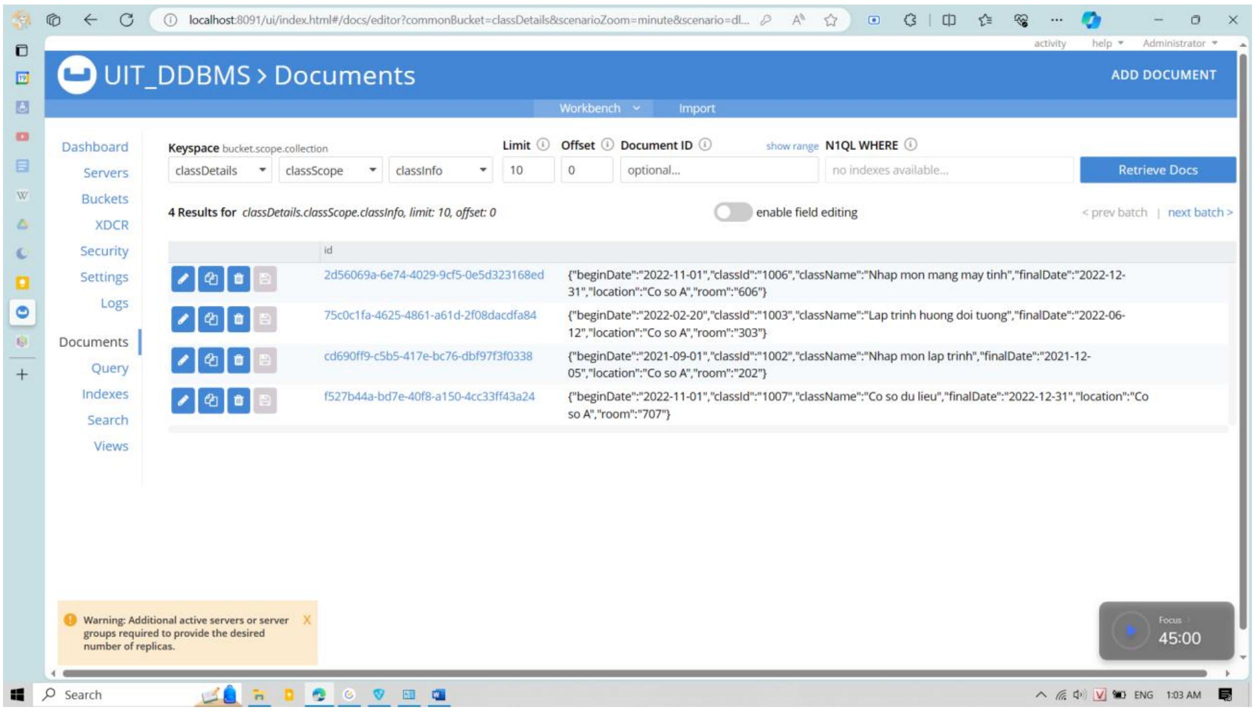
[✓ UUID]

Value of Field:

classId

Picture 3-23

- Back to tab Document to check the data → data was imported successfully

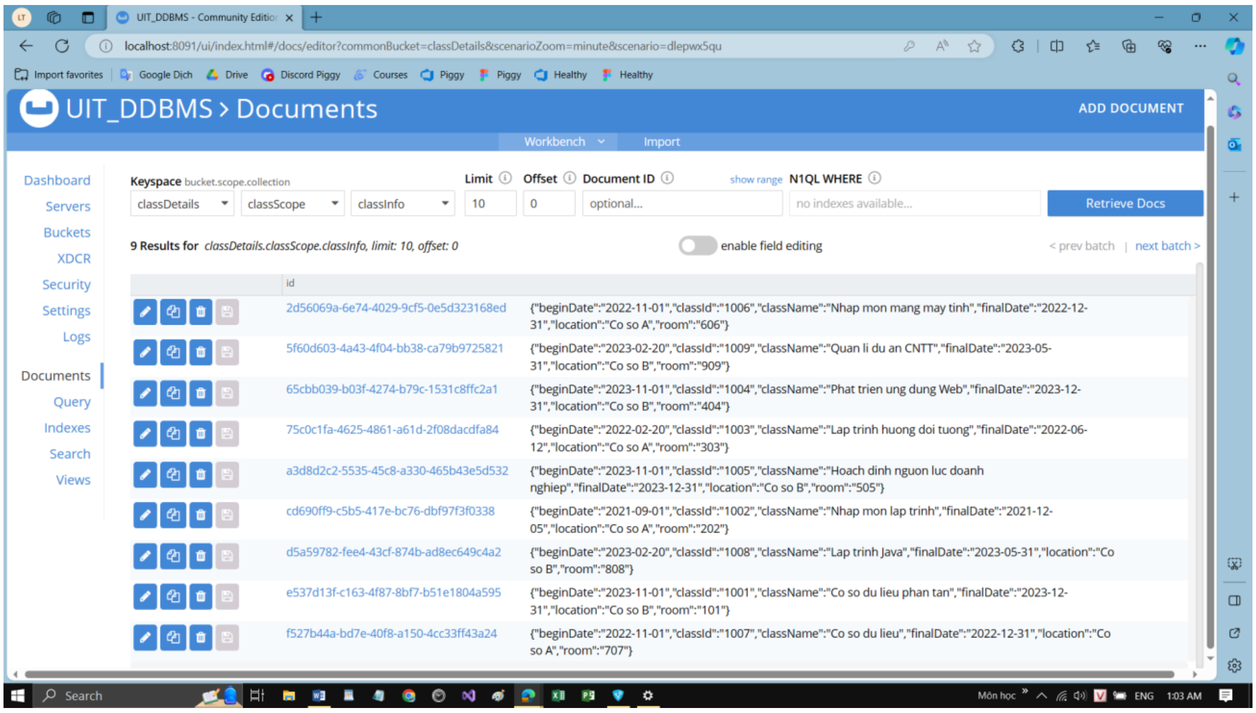


4 Results for *classDetails.classScope.classInfo, limit: 10, offset: 0*

id	
2d56069a-6e74-4029-9cf5-0e5d323168ed	{"beginDate":"2022-11-01","classId":"1006","className":"Nhap mon mang may tinh","finalDate":"2022-12-31","location":"Co so A","room":"606"}
75c0c1fa-4625-4861-a61d-2f08dacdfa84	{"beginDate":"2022-02-20","classId":"1003","className":"Lap trinh huong doi tuong","finalDate":"2022-06-12","location":"Co so A","room":"303"}
cd690ff9-c5b5-417e-bc76-dbf97f3f0338	{"beginDate":"2021-09-01","classId":"1002","className":"Nhap mon lap trinh","finalDate":"2021-12-05","location":"Co so A","room":"202"}
f527b44a-bd7e-40f8-a150-4cc33ff43a24	{"beginDate":"2022-11-01","classId":"1007","className":"Co so du lieu","finalDate":"2022-12-31","location":"Co so A","room":"707"}

Picture 3-24

- Do the same on node B → Check the data

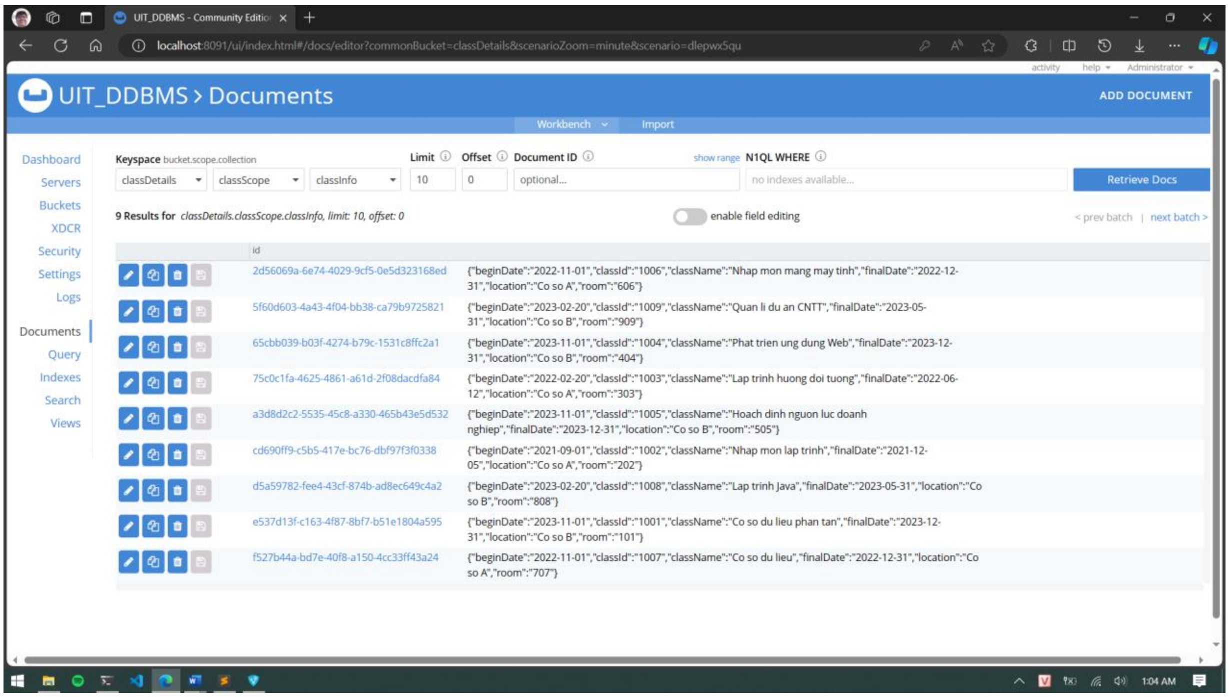


9 Results for *classDetails.classScope.classInfo, limit: 10, offset: 0*

id	
2d56069a-6e74-4029-9cf5-0e5d323168ed	{"beginDate":"2022-11-01","classId":"1006","className":"Nhap mon mang may tinh","finalDate":"2022-12-31","location":"Co so A","room":"606"}
5f60d603-4a43-4f04-bb38-ca79b9725821	{"beginDate":"2023-02-20","classId":"1009","className":"Quan li du an CNTT","finalDate":"2023-05-31","location":"Co so B","room":"909"}
65cbb039-b03f-4274-b79c-1531c8ffc2a1	{"beginDate":"2023-11-01","classId":"1004","className":"Phat trien ung dung Web","finalDate":"2023-12-31","location":"Co so B","room":"404"}
75c0c1fa-4625-4861-a61d-2f08dacdfa84	{"beginDate":"2022-02-20","classId":"1003","className":"Lap trinh huong doi tuong","finalDate":"2022-06-12","location":"Co so A","room":"303"}
a3d8d2c2-5535-45c8-a330-465b43e5d532	{"beginDate":"2023-11-01","classId":"1005","className":"Hoach dinh nguon luc doanh nghiep","finalDate":"2023-12-31","location":"Co so B","room":"505"}
cd690ff9-c5b5-417e-bc76-dbf97f3f0338	{"beginDate":"2021-09-01","classId":"1002","className":"Nhap mon lap trinh","finalDate":"2021-12-05","location":"Co so A","room":"202"}
d5a59782-fee4-43cf-874b-ad8ec649c4a2	{"beginDate":"2023-02-20","classId":"1008","className":"Lap trinh Java","finalDate":"2023-05-31","location":"Co so B","room":"808"}
e537d13f-c163-4f87-8bf7-b51e1804a595	{"beginDate":"2023-11-01","classId":"1001","className":"Co so du lieu phan tan","finalDate":"2023-12-31","location":"Co so B","room":"101"}
f527b44a-bd7e-40f8-a150-4cc33ff43a24	{"beginDate":"2022-11-01","classId":"1007","className":"Co so du lieu","finalDate":"2022-12-31","location":"Co so A","room":"707"}

Picture 3-25

- Check the data on node C



UIT_DDBMS > Documents

Keyspace bucket.scope.collection: classDetails / classScope / classInfo — Limit: 10 — Offset: 0 — Document ID: optional... — N1QL WHERE: no indexes available...

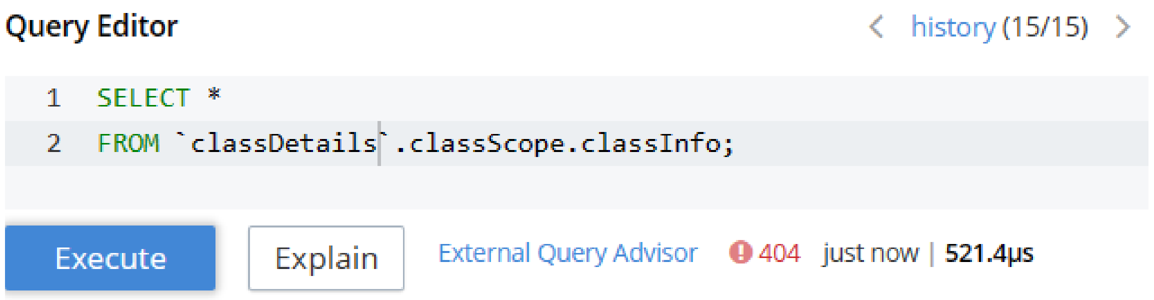
9 Results for *classDetails.classScope.classInfo, limit: 10, offset: 0*

id	
2d56069a-6e74-4029-9cf5-0e5d323168ed	{"beginDate":"2022-11-01","classId":"1006","className":"Nhap mon mang may tinh","finalDate":"2022-12-31","location":"Co so A","room":"606"}
5f60d603-4a43-4f04-bb38-ca79b9725821	{"beginDate":"2023-02-20","classId":"1009","className":"Quan li du an CNTT","finalDate":"2023-05-31","location":"Co so B","room":"909"}
65cbb039-b03f-4274-b79c-1531c8ffc2a1	{"beginDate":"2023-11-01","classId":"1004","className":"Phat trien ung dung Web","finalDate":"2023-12-31","location":"Co so B","room":"404"}
75c0c1fa-4625-4861-a61d-2f08dacdfa84	{"beginDate":"2022-02-20","classId":"1003","className":"Lap trinh huong doi tuong","finalDate":"2022-06-12","location":"Co so A","room":"303"}
a3d8d2c2-5535-45c8-a330-465b43e5d532	{"beginDate":"2023-11-01","classId":"1005","className":"Hoach dinh nguon luc doanh nghiep","finalDate":"2023-12-31","location":"Co so B","room":"505"}
cd690ff9-c5b5-417e-bc76-dbf97f3f0338	{"beginDate":"2021-09-01","classId":"1002","className":"Nhap mon lap trinh","finalDate":"2021-12-05","location":"Co so A","room":"202"}
d5a59782-fee4-43cf-874b-ad8ec649c4a2	{"beginDate":"2023-02-20","classId":"1008","className":"Lap trinh Java","finalDate":"2023-05-31","location":"Co so B","room":"808"}
e537d13f-c163-4f87-8bf7-b51e1804a595	{"beginDate":"2023-11-01","classId":"1001","className":"Co so du lieu phan tan","finalDate":"2023-12-31","location":"Co so B","room":"101"}
f527b44a-bd7e-40f8-a150-4cc33ff43a24	{"beginDate":"2022-11-01","classId":"1007","className":"Co so du lieu","finalDate":"2022-12-31","location":"Co so A","room":"707"}

Picture 3-26

3.2. Query data;

- We will query on node C,

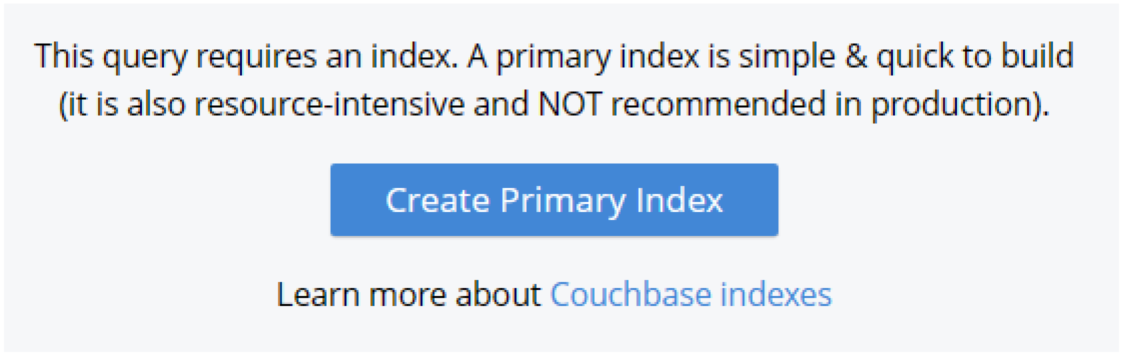


Query Editor

```
SELECT *
FROM `classDetails`.classScope.classInfo;
```

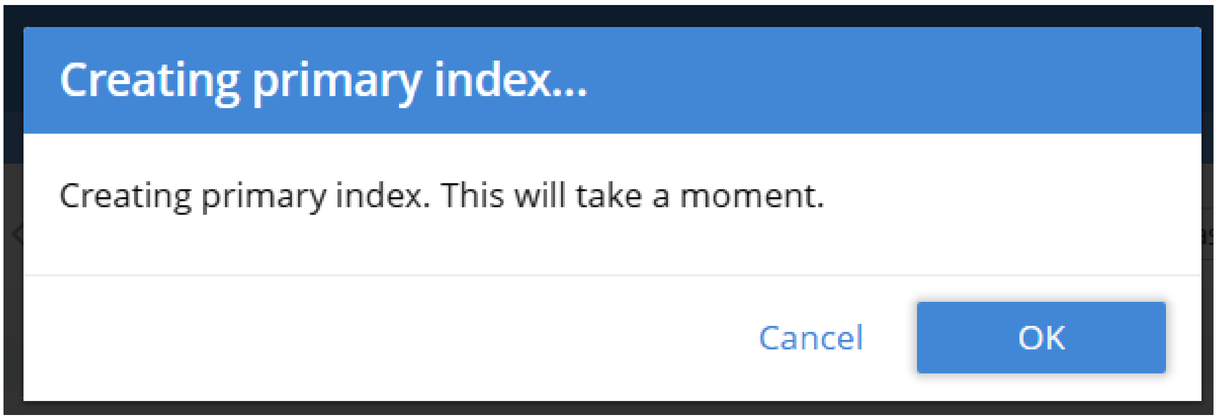
Picture 3-27

- Create Index at first



This query requires an index. A primary index is simple & quick to build (it is also resource-intensive and NOT recommended in production).

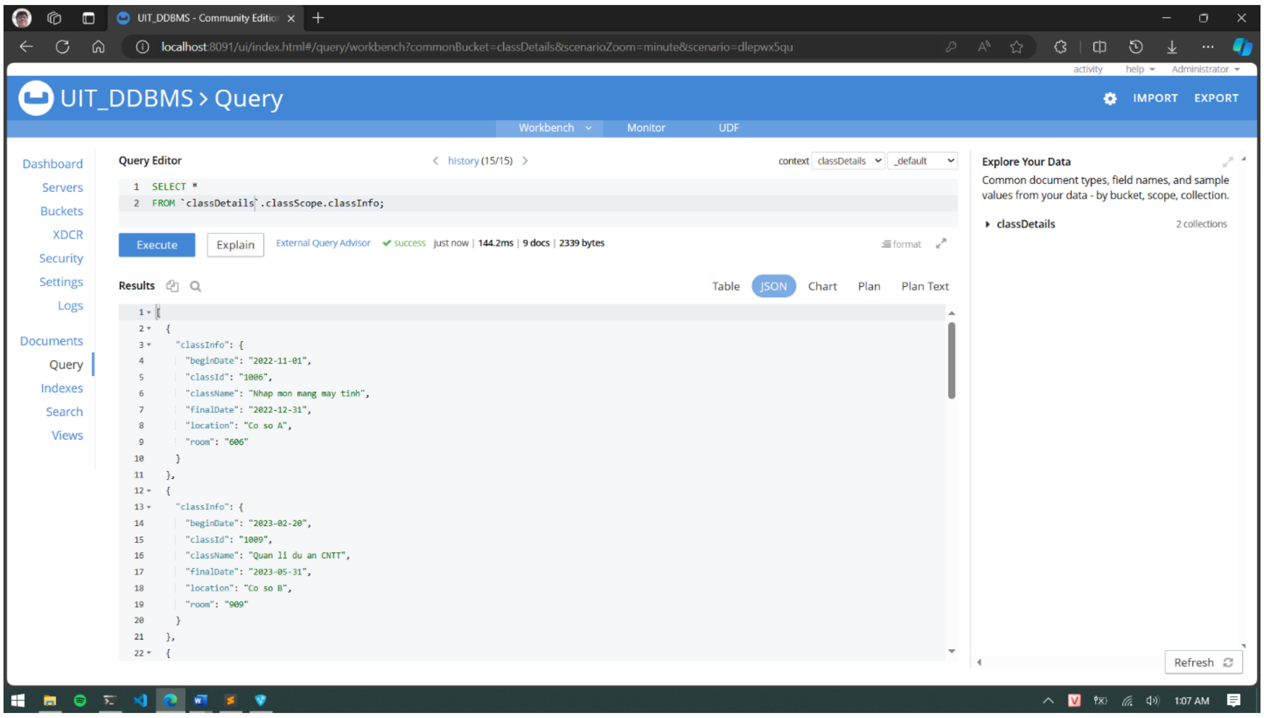
Picture 3-28


Creating primary index...

Creating primary index. This will take a moment.

Picture 3-29

- Try again


localhost:8091/ui/index.html#/query/workbench?commonBucket=classDetails&scenarioZoom=minute&scenario=dlepwx5qu

```
SELECT *
FROM `classDetails`.classScope.classInfo;
```

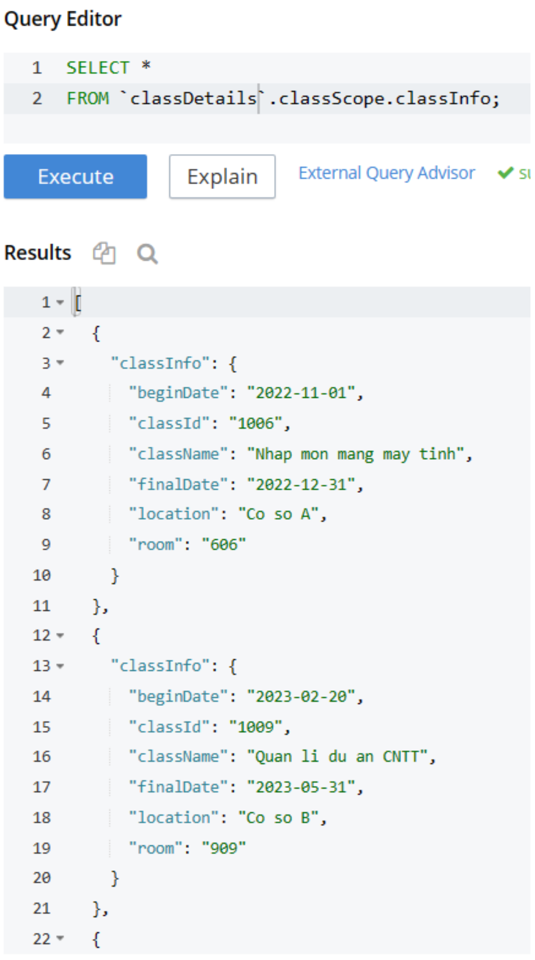
success just now | 144.2ms | 9 docs | 2339 bytes

Common document types, field names, and sample values from your data - by bucket, scope, collection.

```
[
  {
    "classInfo": {
      "beginDate": "2022-11-01",
      "classId": "1006",
      "className": "Nhap mon mang may tinh",
      "finalDate": "2022-12-31",
      "location": "Co so A",
      "room": "606"
    }
  },
  {
    "classInfo": {
      "beginDate": "2023-02-20",
      "classId": "1009",
      "className": "Quan li du an CNTT",
      "finalDate": "2023-05-31",
      "location": "Co so B",
      "room": "909"
    }
  },
  {
```

Picture 3-30

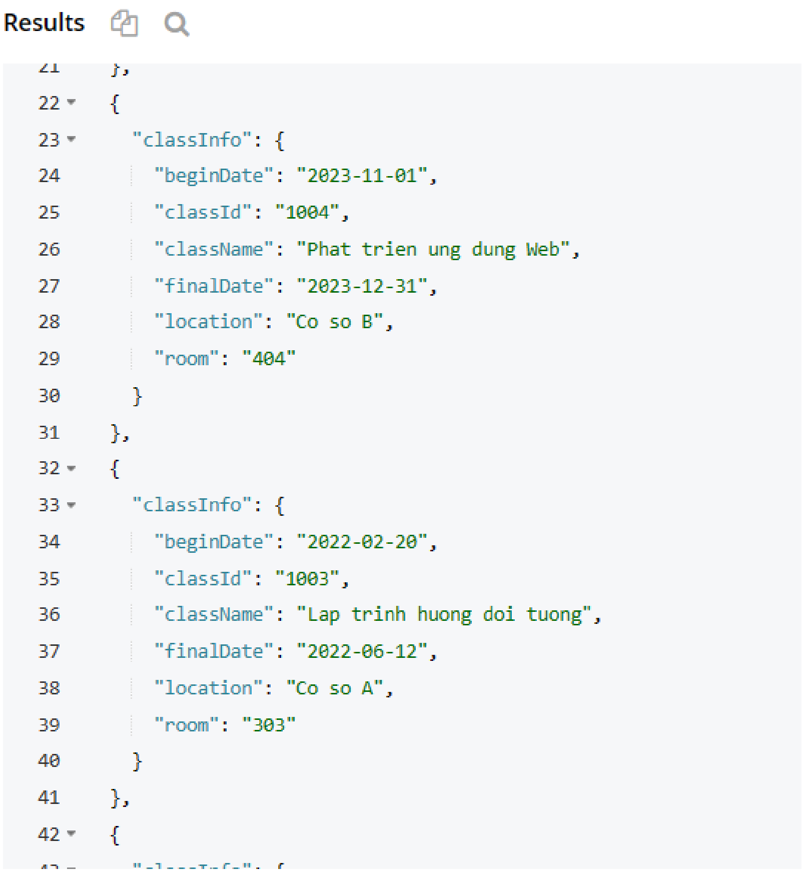
- Result



```
SELECT *
FROM `classDetails`.classScope.classInfo;
```

```
[
  {
    "classInfo": {
      "beginDate": "2022-11-01",
      "classId": "1006",
      "className": "Nhap mon mang may tinh",
      "finalDate": "2022-12-31",
      "location": "Co so A",
      "room": "606"
    }
  },
  {
    "classInfo": {
      "beginDate": "2023-02-20",
      "classId": "1009",
      "className": "Quan li du an CNTT",
      "finalDate": "2023-05-31",
      "location": "Co so B",
      "room": "909"
    }
  },
  {
```

Picture 3-31


Results

```
},
{
  "classInfo": {
    "beginDate": "2023-11-01",
    "classId": "1004",
    "className": "Phat trien ung dung Web",
    "finalDate": "2023-12-31",
    "location": "Co so B",
    "room": "404"
  }
},
{
  "classInfo": {
    "beginDate": "2022-02-20",
    "classId": "1003",
    "className": "Lap trinh huong doi tuong",
    "finalDate": "2022-06-12",
    "location": "Co so A",
    "room": "303"
  }
},
{
```

Picture 3-32

References

[Couchbase Server - Wikipedia]

[Couchbase Server | Couchbase Docs]

[Couchbase - Database of Databases (dbdb.io)]

-