CS300 Couchbase NoSQL Server Administration

July 26th , 2016

Couchbase

Learning Services

Labs – Day 2

Lab #3 (Grow cluster to 6 nodes):

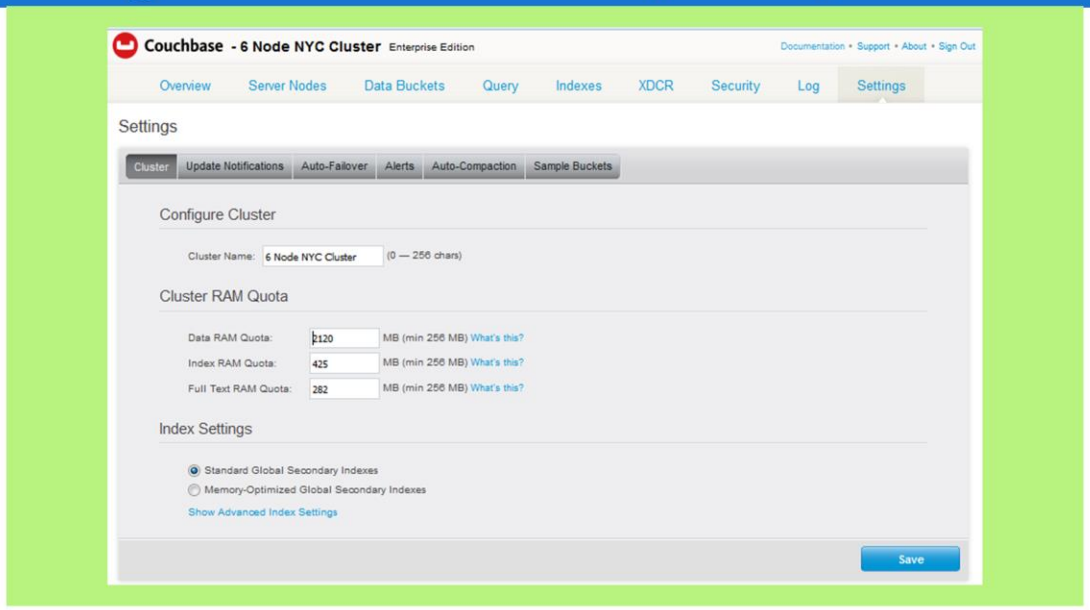
- Exploring the cluster map
- Buckets vs. vBuckets
- Rebalancing
- Introduction to performance metrics via Web UI
- Learn how to adjust the number of replicas
- cbstats command to see active + replica vBuckets

Lab #4 (Removing nodes and MISC commands):

- More details on the Web UI's performance metrics
- Learn how to delete Couchbase buckets
- Gracefully decommission & recommission nodes from a cluster
- Failing over a node in the cluster
- Replica management and deeper understanding
- Using the REST API to check auto-failover settings

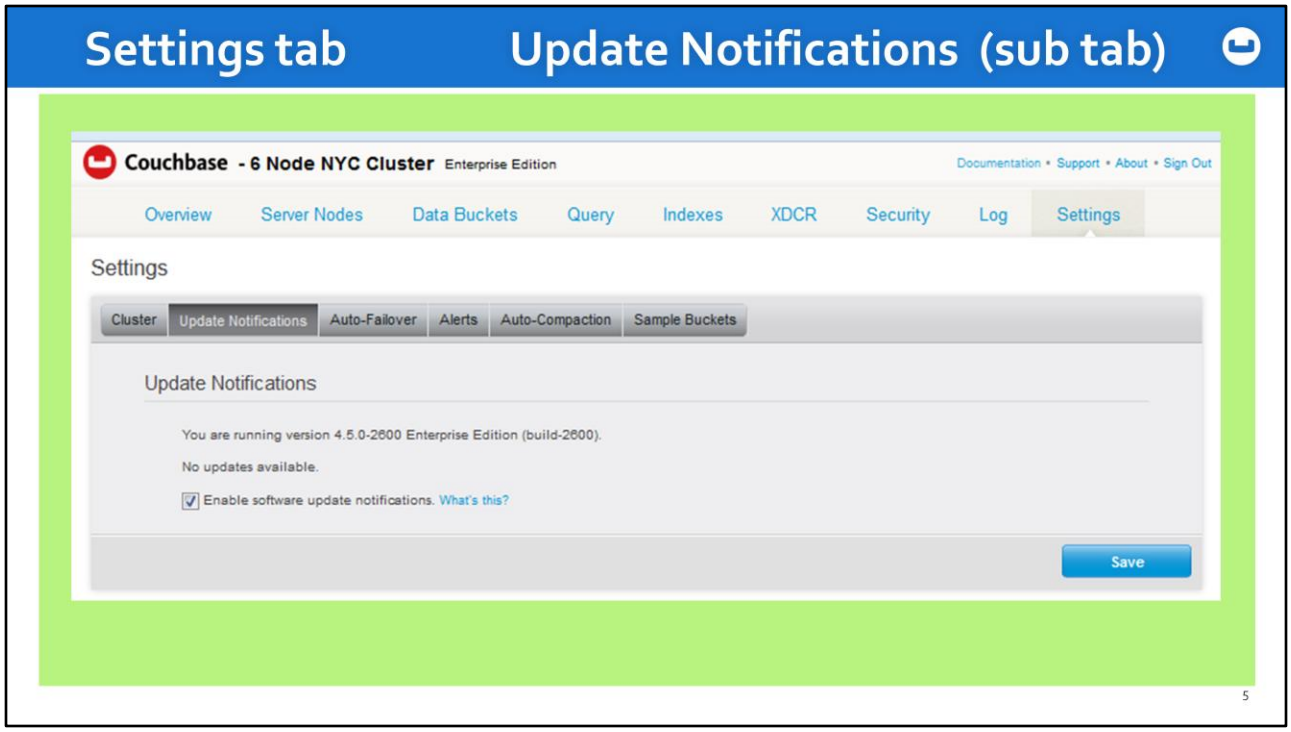
Settings menu in Web UI

Settings tab Cluster (sub tab)



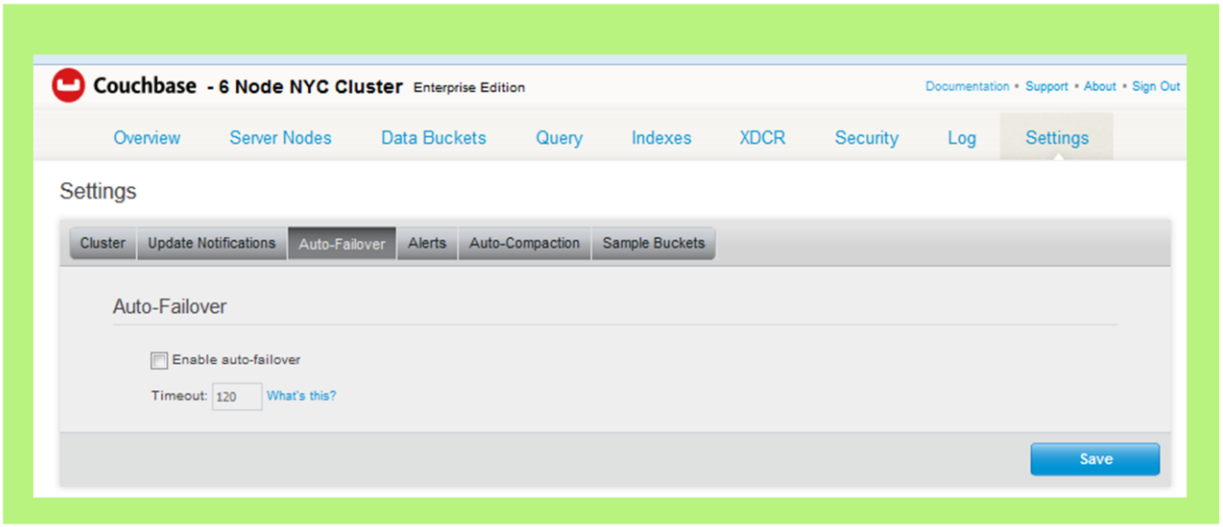
4

Cluster settings show the available RAM on your cluster and the per server RAM quotas.
The Per Server RAM Quota is adjustable and can be set for Data service and for Index service.
In addition, if you have the Enterprise Edition, a Couchbase Server self-signed SSL certification is provided to set up secure communication in an XDCR environment.
The SSL certificate can be regenerated.
Indexer threads can be set for how many are devoted to handling index maintenance.



You can enable or disable Update Notifications by checking the Enable software update notifications checkbox within the Update Notifications screen.
Once you have changed the option, you must click Save to record the change.
If update notifications are disabled then the Update Notifications screen will only display your currently installed version, and no alert will be provided.

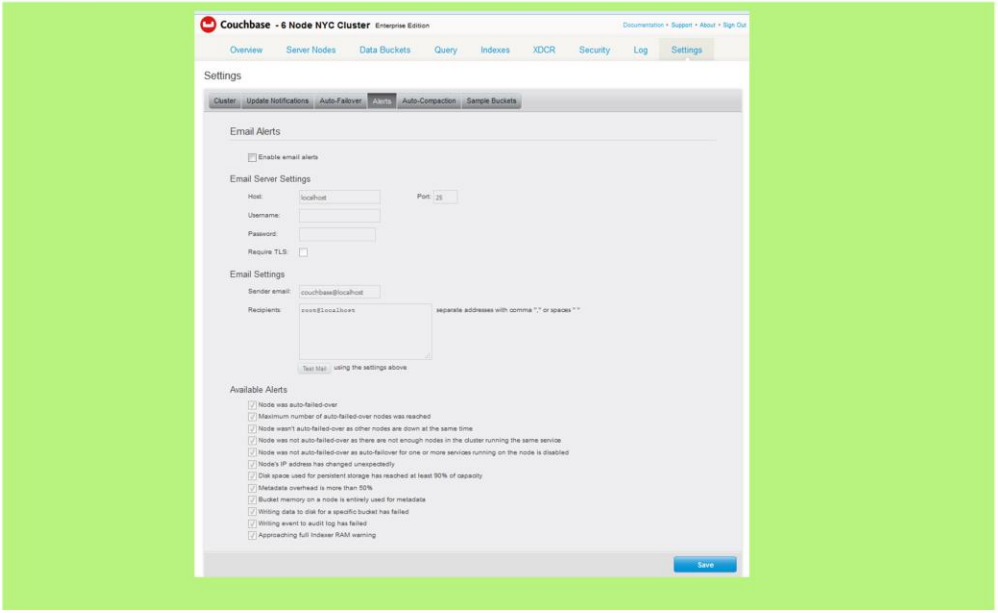
Settings tab Auto-Failover (sub tab)



6

The Auto-Failover settings enable auto-failover, and the timeout before the auto-failover process is started when a cluster node failure is detected.
To enable Auto-Failover, check the Enable auto-failover checkbox.
To set the delay, in seconds, before auto-failover is started, enter the number of seconds it the Timeout box.
The default timeout is 120 seconds. Short time out settings can actually be triggered by network flapping so adjust accordingly.

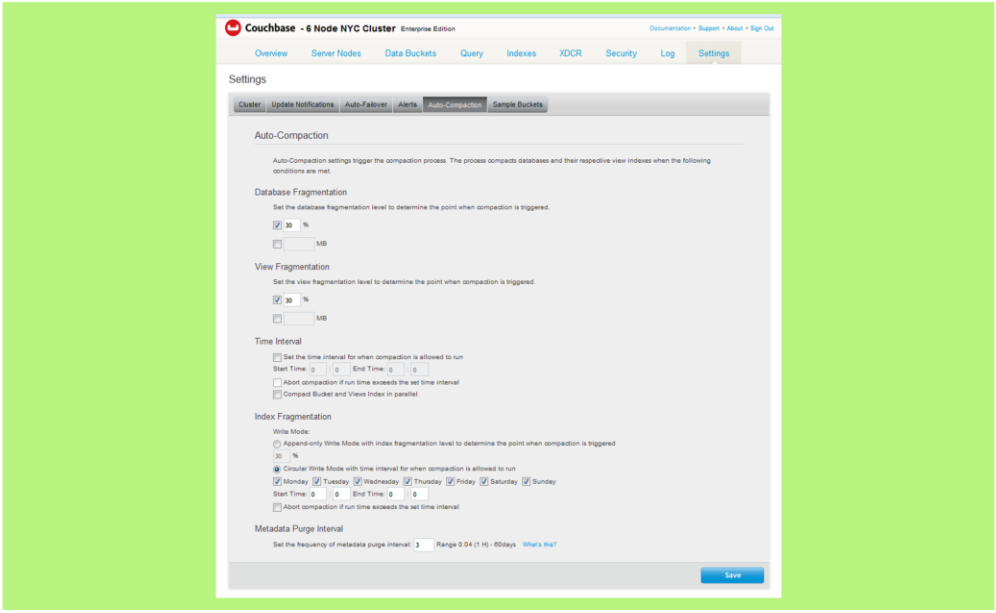
Settings tab Alerts (sub tab)



7

The Alerts can be set to inform of a predetermined set of alert conditions. Email server must be set and interested parties(email) declared.

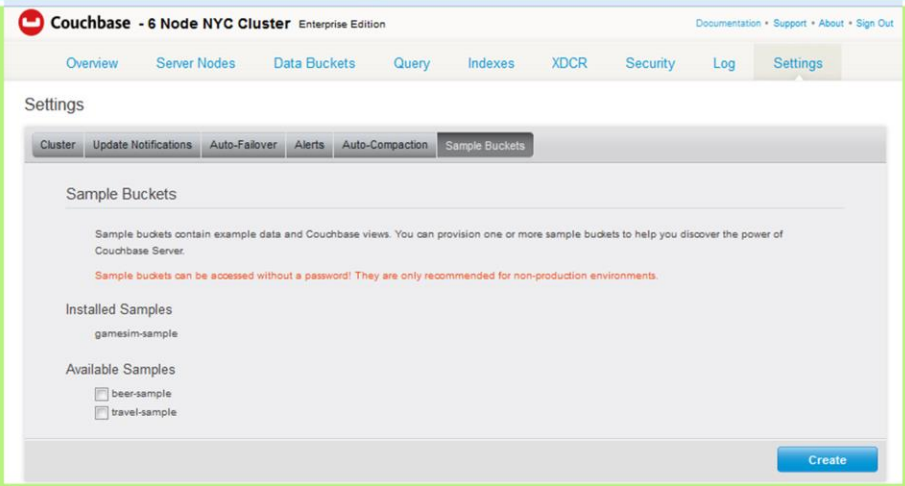
Settings tab Auto-Compaction (sub tab)



8

Auto compaction can be set as a percentage or size(in MB) in the Database, Indexes, and Views.
Additionally it can be set on a system wide schedule on the Auto compaction tab.
Metadata purge interval can also be set here.

Settings tab Sample Buckets (sub tab)

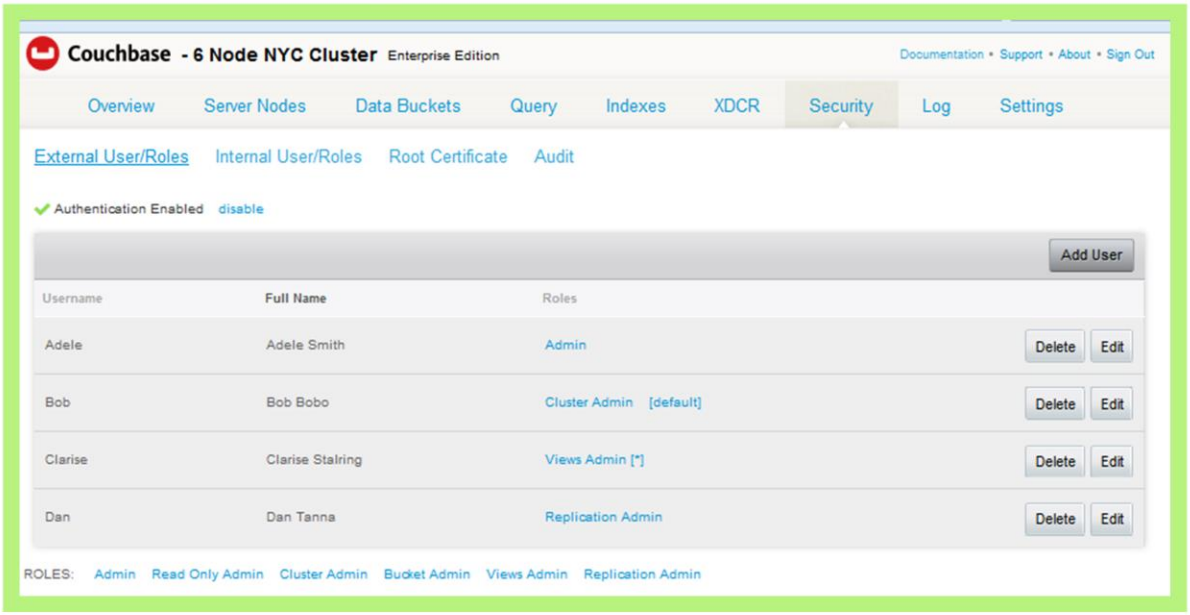



```
# ls  /opt/couchbase/samples
beer-sample.zip  gamesim-sample.zip  travel-sample.zip
```

9

Sample buckets can be loaded with attendant indexes and views.
These can be deleted and reloaded later should the need for development arise.
If you are doing a rebalance you may wish to delete these to speed the rebalance and then load them later.
Directory for including your own sample databases is given here.

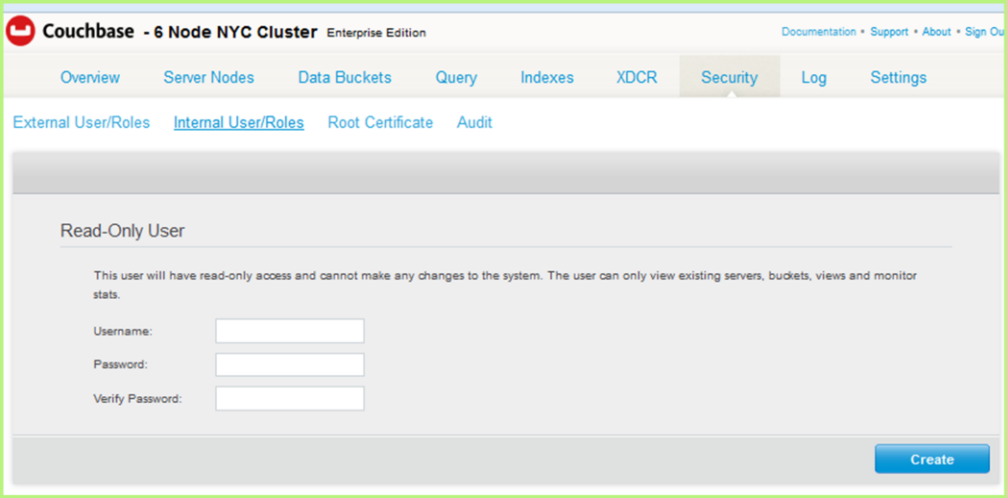
Security Settings tab (sub tab)



10

LDAP authentication can be tied into Couchbase on this tab.
Please read couchbase documentation for the instructions for this implementation.
There are currently only two roles within Couchbase , Full Administrator and Read-Only. LDAP authentication will only map autenticated users to one of these two roles.
In the future there may be more defined roles within Couchbase server.

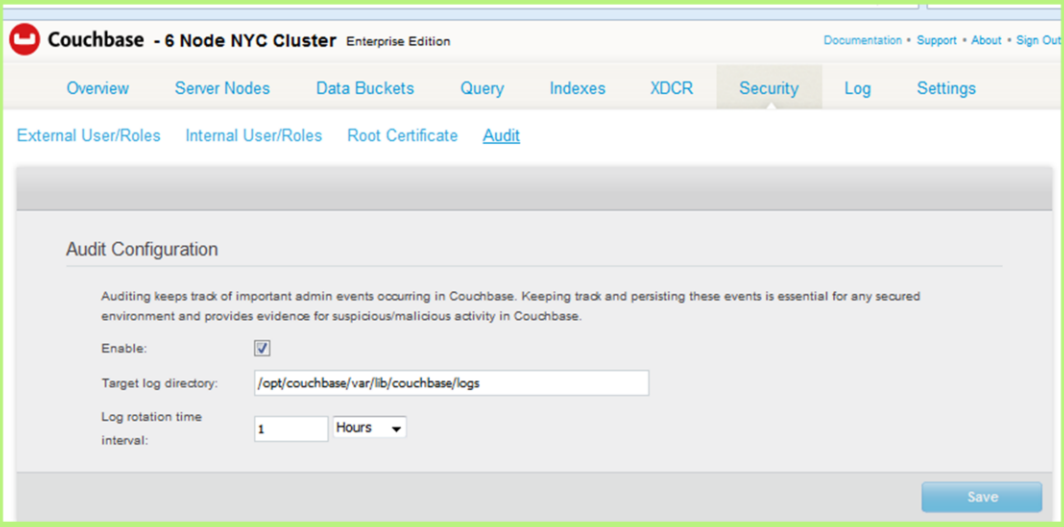
Security tab Account Management (sub tab)



11

Read only account is set up here.

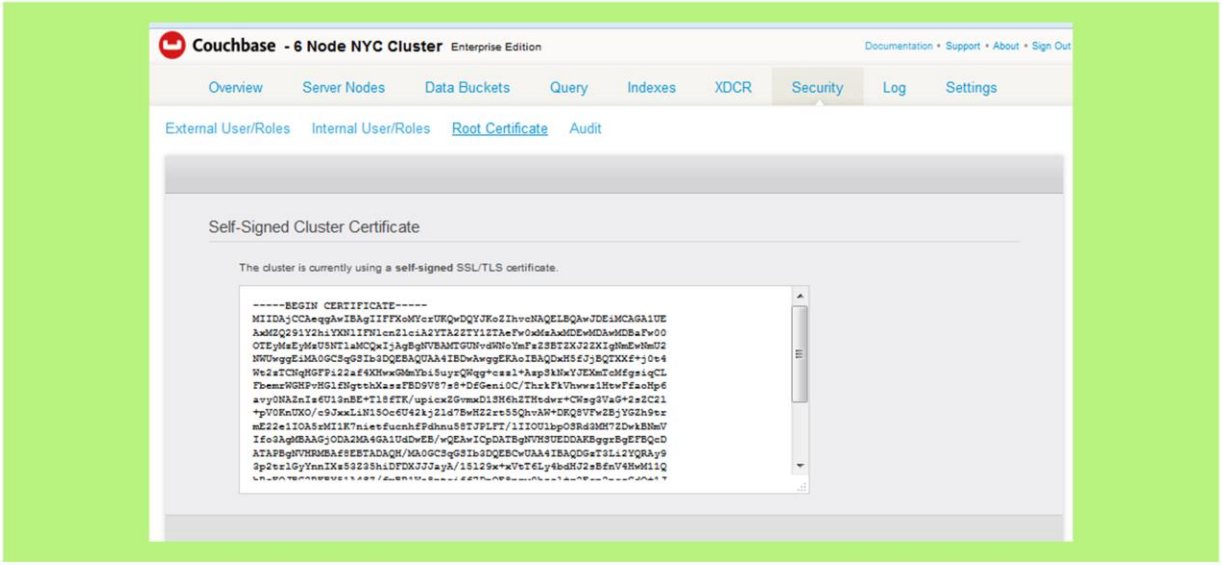
Security tab Audit (sub tab)



12

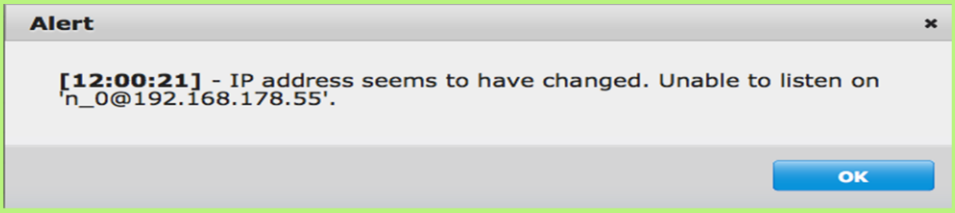
Setting up auditing allows the Administrator to determine who does what .

Security tab Root Certificate (sub tab)



Setting up auditing allows the Administrator to determine who does what .

IP address change alert



14

A new alerting systems has been built into the Couchbase Web Console. This is sued to highlight specific issues and problems that you should be aware of and may need to check to ensure the health of your Couchbase cluster.

Alerts are provided as a popup within the web console. A sample of the IP address popup is shown above:
This may be seen when running couchbase within the cloud with nonpermanent addresses.

The following errors and alerts are supported:

IP Address Changes
If the IP address of a Couchbase Server in your cluster changes, you will be warned that the address is no longer available. You should check the IP address on the server, and update your clients or server configuration.

OOM (Hard) Indicates if the bucket memory on a node is entirely used for metadata.

Commit Failure
Indicates that writing data to disk for a specific bucket has failed.

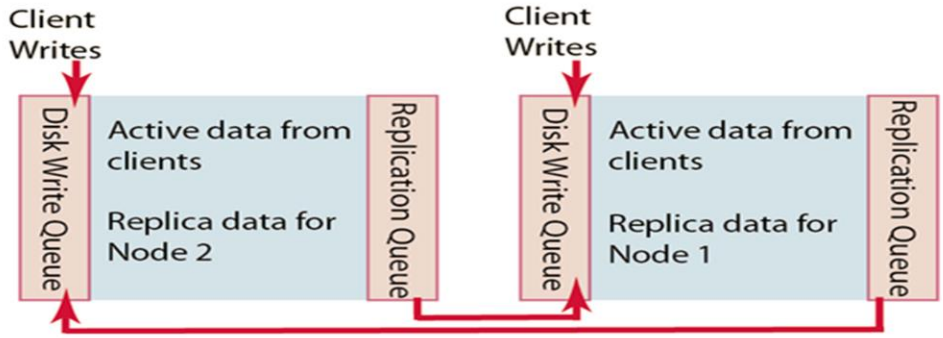
Metadata Overhead
Indicates that a bucket is now using more than 50% of the allocated RAM for storing metadata and keys, reducing the amount of RAM available for data values.

Disk Usage
Indicates that the available disk space used for persistent storage has reached at least 90% of capacity.

15

Replication within a cluster

Replication Details



- If too many items in Disk Write Queue a *backoff message* is sent to the other node
- Other node will then reduce the rate at which it sends items for replication to the complaining node
- If multiple changes occur to the same document waiting to be replicated, Couchbase Server is able to *de-duplicate, or 'de-dup'* the item; this means for the sake of efficiency, it will only send the latest version of a document to the second node.

17

Within a Couchbase cluster, you have *replica data* which is a copy of an item at another node. After you write an item to Couchbase Server, it makes a copy of this data from the RAM of one node to another node. Distribution of replica data is handled in the same way as active data; portions of replica data will be distributed around the Couchbase cluster onto different nodes to prevent a single point of failure. Each node in a cluster will have *replica data* and *active data* ; replica data is the copy of data from another node while active data is data that had been written by a client on that node.

Replication of data between nodes is entirely peer-to-peer based; information will be replicated directly between nodes in the cluster. There is no topology, hierarchy or master-slave relationship between nodes in a cluster. When a client writes to a node in the cluster, Couchbase Server stores the data on that node and then distributes the data to one or more nodes within a cluster. The following shows two different nodes in a Couchbase cluster, and illustrates how two nodes can store replica data for one another:

When a client application writes data to a node, that data will be placed in a replication queue and then a copy will be sent to another node. The replicated data will be available in RAM on the second node and will be placed in a disk write queue to be stored on disk at the second node.

Notice that a second node will also simultaneously handle both replica data and incoming writes from a client. The second node will put both replica data and incoming writes into a disk write queue. If there are too many items in the disk write queue, this second node can send a *backoff message* to the first node. The first node will then reduce the rate at which it sends items to the second node for replication. This can sometimes be necessary if the second node is already handling a large volume of writes from a client application.

Replication Details

- Each bucket can have a different # of replicas (max = 3, so 4 total copies of data)
- Even if you configure three replicas for a data bucket, replication will only be enabled once you have four nodes in the cluster
- On all nodes, both active and replica data must wait in a disk write queue before being written to disk
- By default a node will send backoff messages when the disk write queue on the node contains one million items or 10% of total items in replica partition. Other nodes will then reduce the rate at which they send replica data
- You can configure this default to be a given number so long as this value is less than 10% of the total items currently in a replica partition.
- Example: node contains 20 million items, when the disk write queue reaches 2 million items, a backoff message will be sent to nodes sending replica data.

18

Replication Details

To change the "backoff messages" setting to 2 million or 15% of all items (whichever is greater):

```
# /opt/couchbase/bin/cbepctl  10.5.2.31:11210 -b
bucket_name -p bucket_password  set  dcp_param
dcp_throttle_queue_cap  2000000
```

setting param: dcp_throttle_queue_cap 2000000
or

```
# /opt/couchbase/bin/cbepctl 10.5.2.31:11210 set -b
bucket_name dcp_param dcp_throttle_cap_pcnt 15
```

Warning: this tool is a per-node, per-bucket operation

19

In the first example we specify that a node sends replication backoff requests when it has two million items or 10% of all items, whichever is greater.

In this next example, we change the default percentage used to manage the replication stream. If the items in a disk write queue reach the greater of this percentage or a specified number of items, replication requests will slow down. In this example, we set the threshold to 15% of all items at a replica node. When a disk write queue on a node reaches this point, it will send replication backoff requests to other nodes.

Be aware that this tool is a per-node, per-bucket operation. That means that if you want to perform this operation, you must specify the IP address of a node in the cluster and a named bucket. If you do not provided a named bucket, the server will apply the setting to any default bucket that exists at the specified node. If you want to perform this operation for an entire cluster, you will need to perform the command for every node/bucket combination that exists for that cluster.

TAP Queues (deprecated)

20

Tap queues will be deprecated and are not used after all node in the cluster have been upgraded to 3.0 or above.

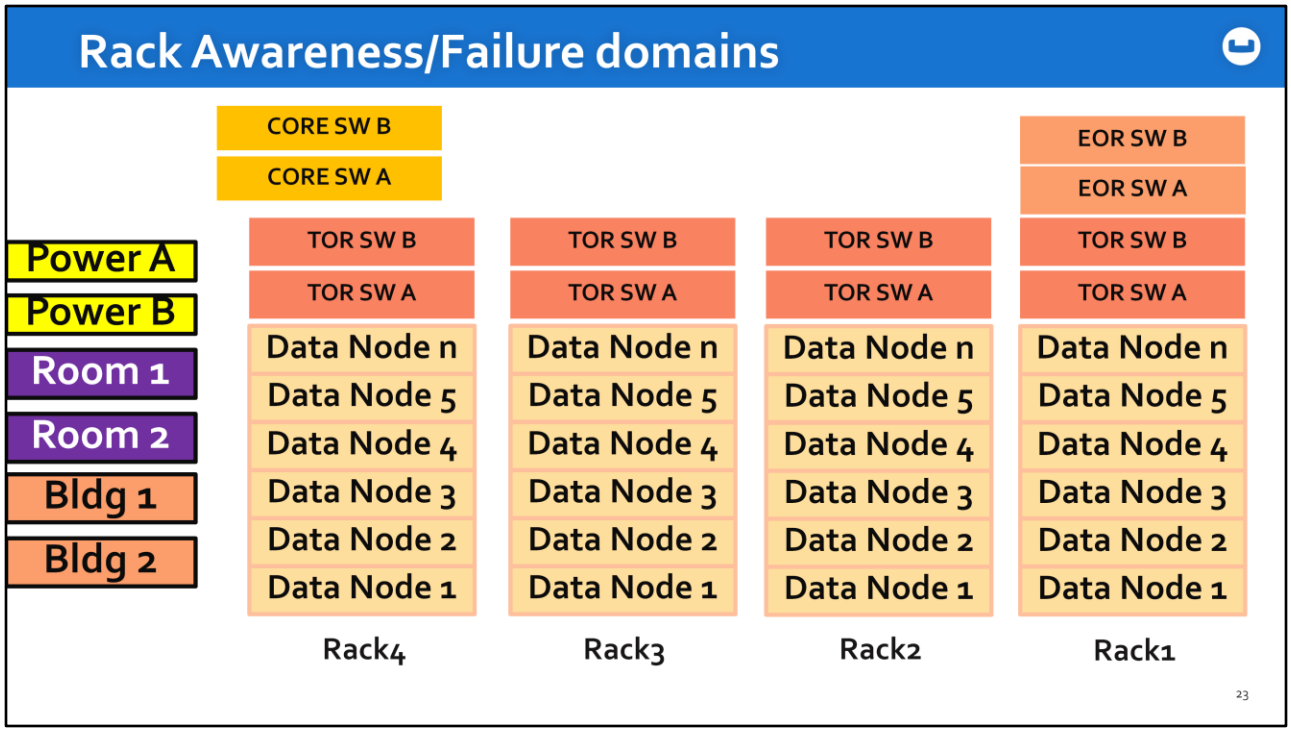
backfill remaining
Number of items in the backfill queue for the corresponding TAP connection for this bucket.

DCP Queues (introduced in 3.0)

21

DCP Queues offer much that same in the away of metrics to be monitored. They are colunmized by Replication, XDCR, Views indexes and other.

Rack Awareness



Failure domains can be any logical grouping.
Power
Rack
Row
Room
Floor
Bldg
Etc…

Failure domains are logically grouped in couchbase by “Group Number” i.e. Server Groups

Allows you to define failure points in the environment. A rack, switch, hypervisor, zone…..
Works by organizing vbuckets across nodes so that a entire group of servers can be failed over regardless of replica count.
Servers can either be manually rebalanced in and assigned to a group OR added afterwards.
If adding afterwards, a rebalance is MANDATORY

Rack Awareness prerequisites

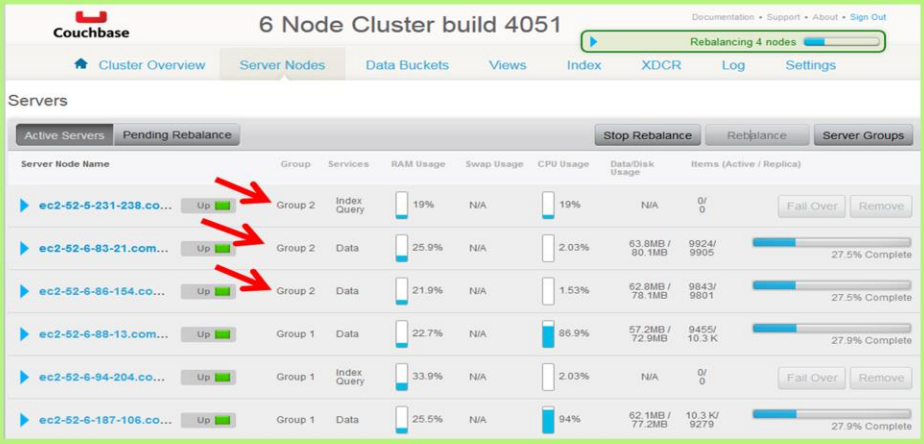
At least Couchbase 2.5 Enterprise Edition

Configure at least two server groups.

Configure all of the servers to use server groups.

Configure each server group to have the same number of servers (recommended).

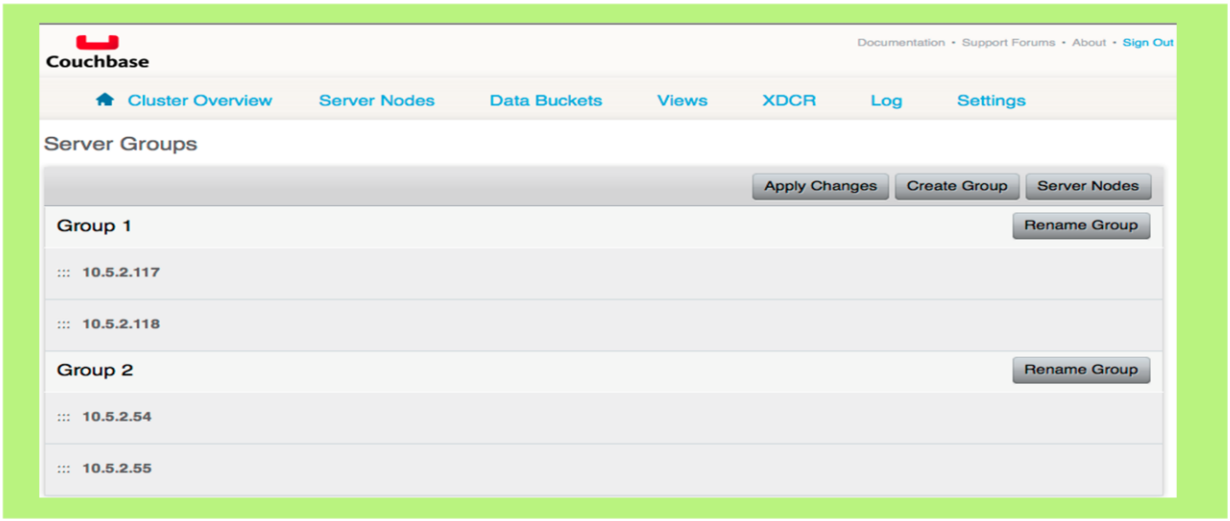
Server Groups



24

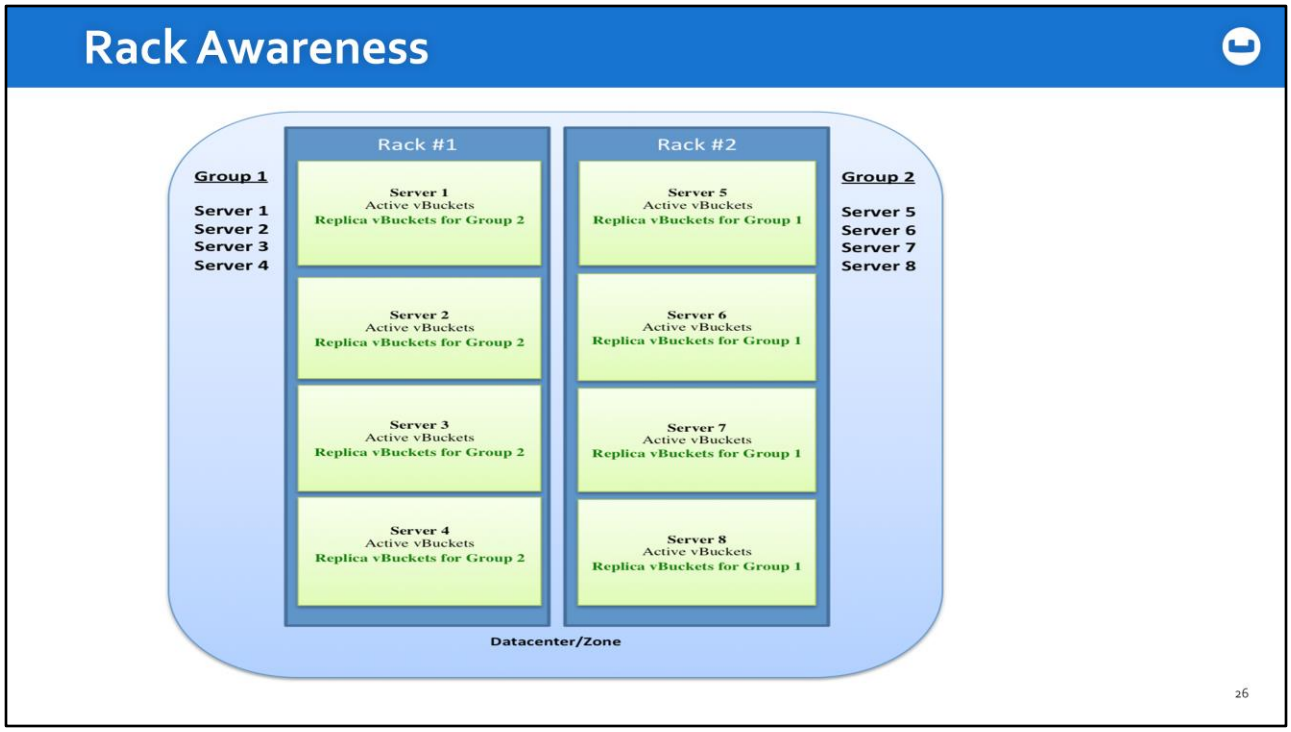
Balanced server services .

Server Groups view



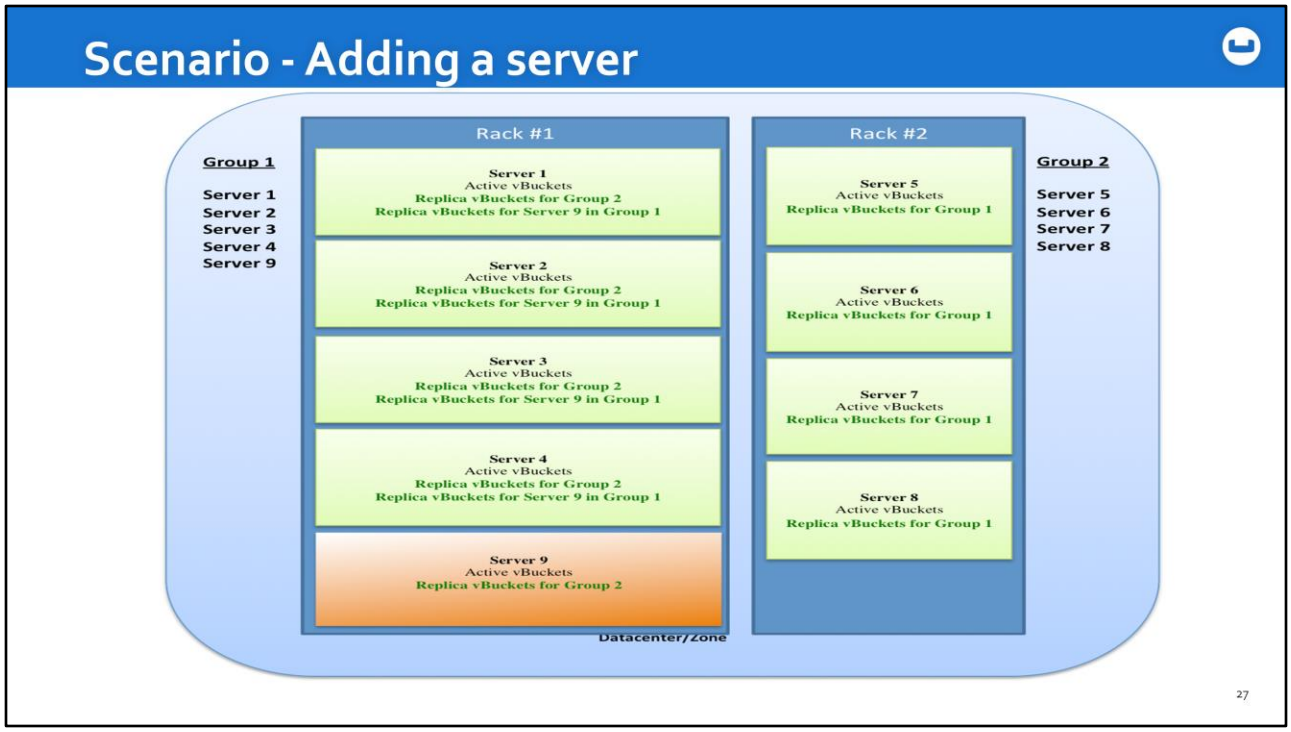
25

Nodes can be moved from groups after creation by drag and drop into new group.
Hit apply changes button after making any changes.



The following example shows how Rack Awareness functionality implements replica vBuckets to provide redundancy. In this example, there are two (2) server groups in the cluster and four (4) servers in each server group. Since there is equal number of servers in each server group, the cluster is balanced which guarantees that replica vBuckets for one server group are on a different server group.

The following diagram shows a cluster of servers on two racks, Rack #1 and Rack #2, where each rack has a group of four (4) servers.
Group 1 has Servers 1, 2, 3, and 4.
Group 1 servers have their active vBuckets and replica vBuckets from Group 2.
Group 2 has Servers 5, 6, 7, and 8.
Group 2 servers have their active vBuckets and replica vBuckets from Group 1.



The following scenario shows how Rack Awareness functionality implements replica vBuckets when an imbalance is caused by an additional server being added to one server group. In this example, an additional server (Server 9) is added to a server group (Group 1). An imbalance occurs because one server group has more servers than the other server group. In this case, the rebalance operation performs a “best effort” of evenly distributing the replica vBuckets of the additional server across the nodes on all the racks in the cluster.

The following diagram shows a cluster of servers on two racks, Rack #1 and Rack #2, where one rack has a group of five (5) servers and the other rack has a group of four (4) servers.

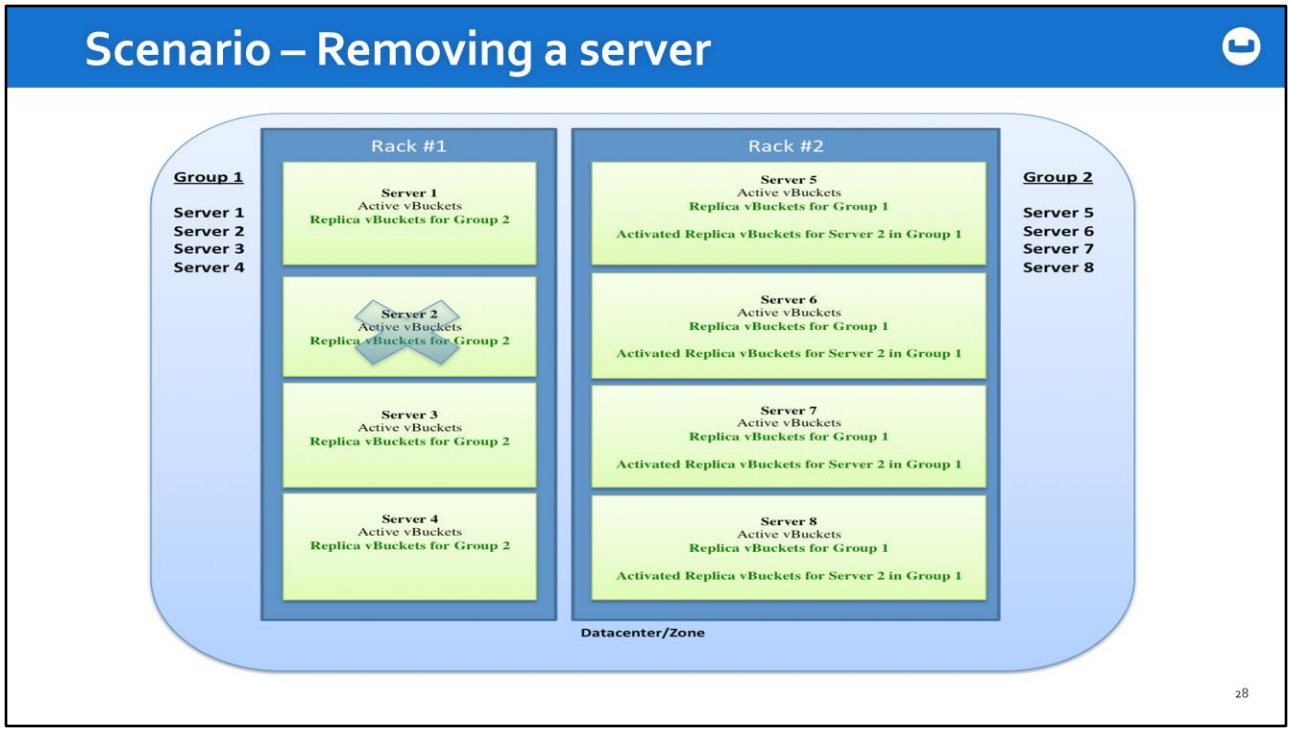
Group 1 has Servers 1, 2, 3, 4, and 9

Group 1 servers have their active vBuckets and replica vBuckets from Group 2.

Group 1 Servers 1 - 4 also has replica vBuckets for Server 9.

Group 2 has Servers 5, 6, 7, and 8.

Group 2 servers have their active vBuckets and replica vBuckets from Group 1 including the replica vBuckets from Server 9 in Group 1.



The following diagram shows the loss of a server resulting in an imbalance. In this case, Server 2 (from Group 1, Rack #1) becomes unavailable. The replica vBuckets for Server 2 in Group 2, Rack #2 become enabled and rebalancing occurs.

Group 1 has Servers 1, 2, 3, and 4
Group 1 servers have their active vBuckets and replica vBuckets from Group 2.
Group 1 Server 2 becomes unavailable.
Group 2 has Servers 5, 6, 7, and 8.
Group 2 servers have their active vBuckets and replica vBuckets from Group 1.
Group 2 server activates the replica vBuckets for Server 2 in Group 1.

Lab #3: Grow cluster to 6 nodes

- Exploring the cluster map

Time: 1-1.5 hour

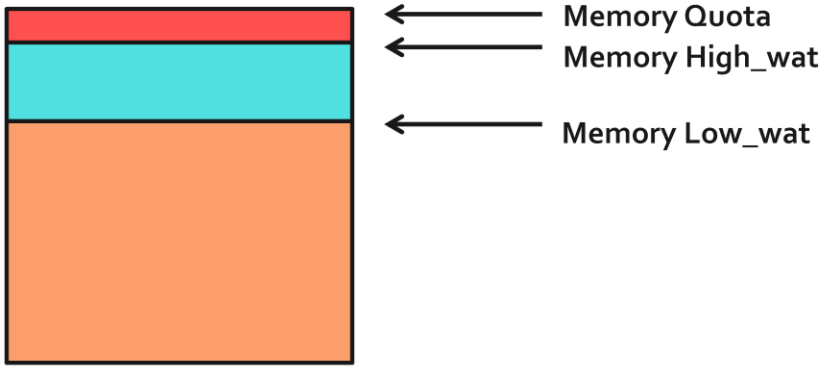
- Buckets vs. vBuckets
- Rebalancing
- Introduction to performance metrics via Web UI
- Adjust the number of replicas
- Use the cbstats command to see active + replica vBuckets

29

Data Ejection and Working Set management

High and low water marks

Bucket Configuration



Working Set: Keys, metadata & frequently accessed items

All Keys and metadata are always kept in the working set!

31

keys & metadata in RAM serves 3 important purposes

- **Key and metadata in RAM are used are used to fulfill a read request from a client by acknowledging that a key indeed exists using RAM metadata, then fetching the item from disk**
- **Miss access: ability to quickly determine whether a key exists or not (used for adds/replace which don't actually require the item to be read)**
- **Expiry pager: the expiration process in Couchbase Server uses the metadata in RAM to quickly scan for items that are expired and later remove them from disk. Runs every 60 minutes by default**

32

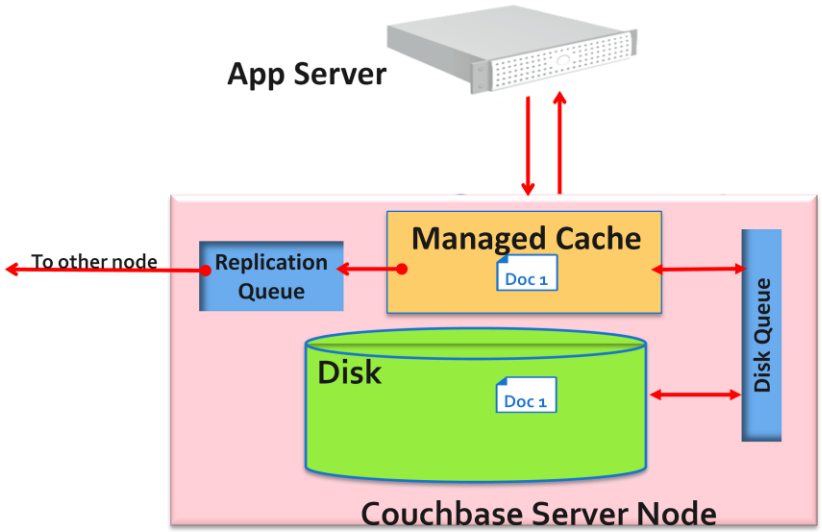
It's not as much about quickly locating the item in disk, but more about _very_ quickly being able to tell whether the item exists at all or not.

With all the metadata and keys in RAM, we can do sub-ms "misses" when a request comes in for a key that doesn't exist...as well as doing sub-ms for some of the atomic requests like add or replace that don't actually require the value to be read.

Read Operations and Cache

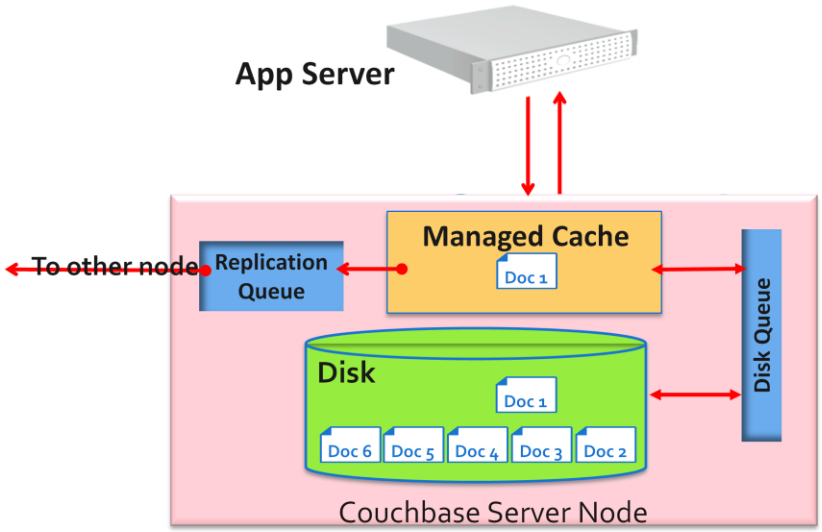
- Get operations request the document
- If the document is in the cache, return it
- If the data is larger than the cache size, NRU items are ejected from cache (assuming they are persisted to disk)
- Document ID and metadata stays in RAM
  - This ensures that the server returns a valid/invalid document ID without waiting for disk access

Couchbase Read Operation



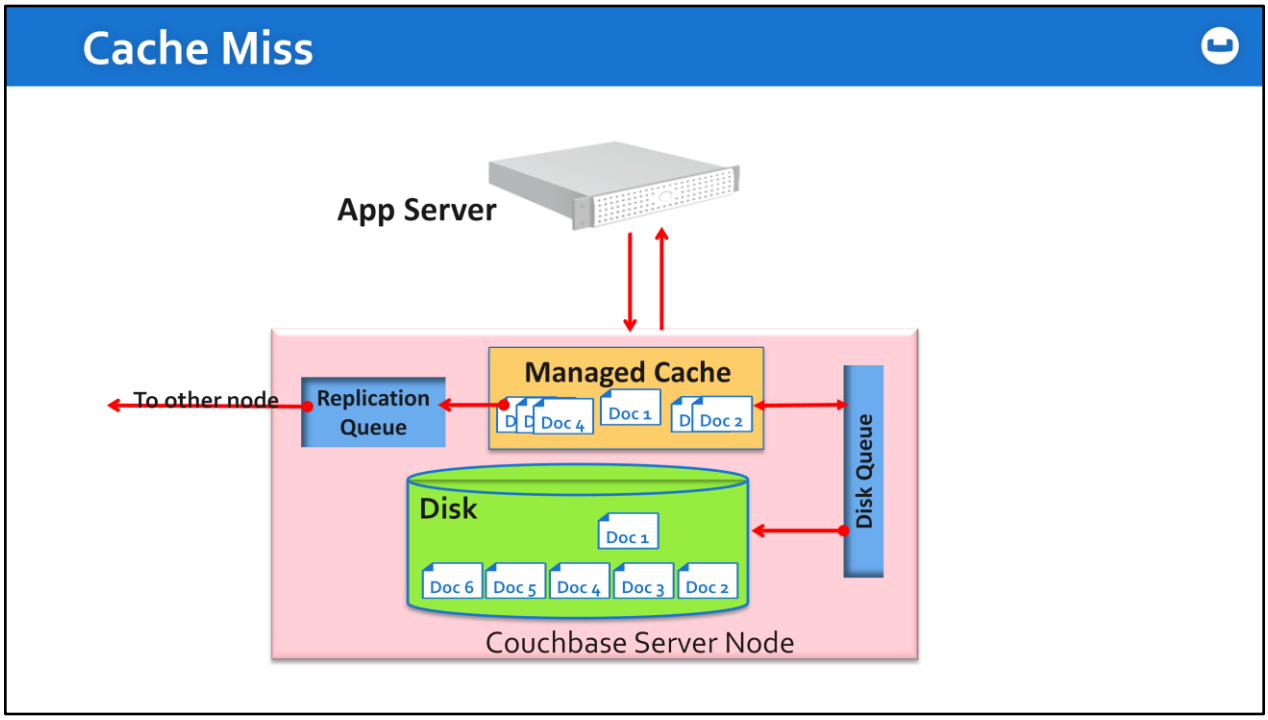
Get operations request the document
If the document is in the cache, return it
If the data is larger than the cache size, NRU items are ejected from cache (assuming they are persisted to disk)
Document ID and metadata stays in RAM
This ensures that the server returns a valid/invalid document ID without waiting for disk access

Cache Ejection



Now, as you fill up memory (click), some data that has **already been written to disk** will be ejected from RAM to make room for new data. (click)

Couchbase supports holding much more data than you have RAM available. It's important to size the RAM capacity appropriately for your working set: the portion of data your application is working with at any given point in time and needs very low latency, high throughput access to. In some applications this is the entire data set, in others it is much smaller. As RAM fills up, we use a "not recently used" algorithm to determine the best data to be ejected from cache.



Should a read now come in for one of those documents that has been ejected (click), it is copied back from disk into RAM and sent back to the application. The document then remains in RAM as long as there is space and it is being accessed.

Not-Frequently-Used Items

NRU: All items in the server contain metadata indicating whether the item has been recently accessed or not (this is called NRU)

NRU items are candidates for ejection after high-water mark is exceeded

NRUs are decremented or incremented by server processes to indicate an item is more frequently used, or less frequently used. Items with lower bit values have lower scores and are considered more frequently used.

37

Not Recently Used (NRU) Score

- **Maintain two-bits long NRU score per item in hash table**
- **NRU score sets to "2" for each new item and gets decremented by "1" for each READ access**

NRU Score	Access Pattern
3 → 2	Accessed by READ
2 → 3	Incremented by Item Pager
2	Initial value for a new item
2 → 1	Accessed by READ
1 → 2	Incremented by Item Pager
1 → 0	Accessed by READ
0 → 1	Incremented by Item Pager
0 → 0	Accessed by READ

This is a day in the life of the epiry pager.
When it comes in contact with documents in varying NRU conditions

Item Pager

- A process that runs periodically, removes documents from RAM and retains the item's key and metadata

Version	High Water Mark	Low Water Mark
2.0	75%	60%
2.0.1 and higher	85%	75%

- If high water mark is reached, both active + replica data is ejected until low water mark

39

Value Ejection: is a process automatically performed by Couchbase Server; it is the process of removing data from RAM to provide room for frequently-used items. When Couchbase Server ejects information, it works in conjunction with the disk persistence system to ensure that data in RAM has been persisted to disk and can be safely retrieved back into RAM if the item is requested. The process that Couchbase Server performs to free space in RAM, and to ensure the most-used items are still available in RAM is also known as working set management.

Full ejection allows ejection of keys and metadata, there will be a penalty in latency should documents and metadata and keys need to be retrieved form disk.

Eviction: Some of you may be using only memcached buckets with Couchbase Server; in this case the server provides only a caching layer as storage and no data persistence on disk. If your server runs out of space in RAM, it will evict items from RAM on a least recently used basis (LRU). Eviction means the server will remove the key, metadata and all other data for the item from RAM. After eviction, the item is irretrievable.

Item Pager

The item pager ejects items from RAM in two phases:

Phase 1: Eject based on NRU. Scan NRU for items and create list of all items with score of 3. Eject all items with a NRU score of 3. Check RAM usage and repeat this process if usage is still above the low water mark.

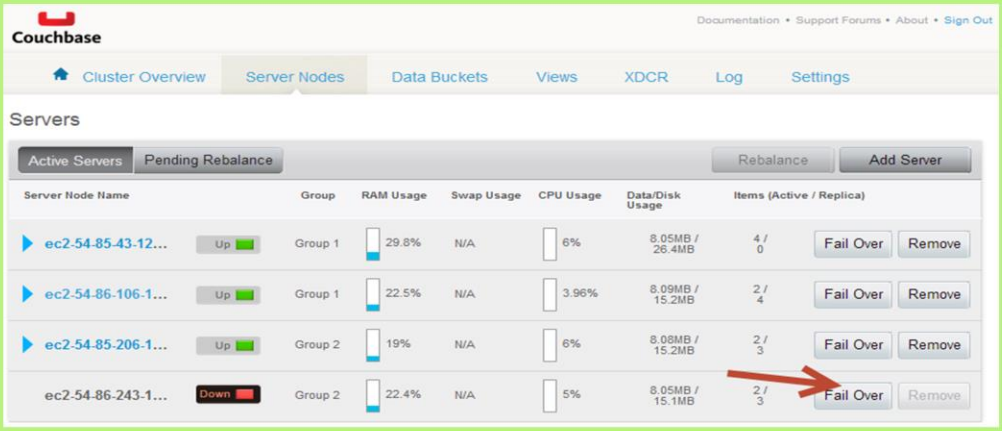
Phase 2: Eject based on Algorithm. Increment all item NRUs by 1. If an NRU is equal to 3, generate a random number and eject that item if the random number is greater than a specified probability. The probability is based on current memory usage, low water mark, and whether a vBucket is in an active or replica state. If a vBucket is in active state the probability of ejection is lower than if the vBucket is in a replica state. Default probabilities:

Evictions of active and replica data occur with the ratio probability of 40% (active data) to 60% (replica data) until the memory usage reaches the low watermark. (can be changed with cbepctl command)

40

Failover

Failover


Couchbase
Documentation • Support Forums • About • Sign Out
Cluster Overview
Server Nodes
Data Buckets
Views
XDCR
Log
Settings
Servers
Active Servers
Pending Rebalance
Rebalance
Add Server
Server Node Name
Group
RAM Usage
Swap Usage
CPU Usage
Data/Disk Usage
Items (Active / Replica)
ec2-54-85-43-12... Up Group 1 29.8% N/A 6% 8.05MB / 26.4MB 4 / 0 Fail Over Remove
ec2-54-86-106-1... Up Group 1 22.5% N/A 3.96% 8.09MB / 15.2MB 2 / 4 Fail Over Remove
ec2-54-85-206-1... Up Group 2 19% N/A 6% 8.08MB / 15.2MB 2 / 3 Fail Over Remove
ec2-54-86-243-1... Down Group 2 22.4% N/A 5% 8.05MB / 15.1MB 2 / 3 Fail Over Remove

42

Failover

- During failover Couchbase removes the node from a cluster and makes replicated data at other nodes available (replica vBuckets made active) for client requests
- It is possible to set automatic-failover (warning: workload for remaining nodes will increase), but it is disabled by default
- Best practice: After failover, return a good/functioning node to cluster and then rebalance for cluster to recover to original state
- Failover NOT = to removing + rebalancing a node!
- Auto-failover still requires a rebalance to make cluster healthy again!

43

When you remove a functioning node from a cluster, you use Web Console to indicate the node will be removed, then you rebalance the cluster so that data requests for the node can be handled by other nodes. Since the node you want to remove still functions, it is able to handle data requests until the rebalance completes(graceful failover).

At this point, other nodes in the cluster will handle data requests.
There is therefore no disruption in data service or no loss of data that can occur when you remove a node then rebalance the cluster.
If you need to remove a functioning node for administration purposes, you should use the remove and rebalance functionality not failover.

After auto-failover

```
# curl -i -u cluster-username:cluster-password
http://localhost:8091/settings/autoFailover/resetCount
```

After a node has been automatically failed over, Couchbase Server increments an internal counter that indicates if a node has been failed over

This counter prevents the server from automatically failing over additional nodes until you identify the issue that caused the failover and resolve it.

If the internal counter indicates a node has failed over, the server will no longer automatically failover additional nodes in the cluster.

You will need to re-enable automatic failover in a cluster by resetting this counter.

44

After a node has been automatically failed over, Couchbase Server increments an internal counter that indicates if a node has been failed over

This counter prevents the server from automatically failing over additional nodes until you identify the issue that caused the failover and resolve it.

If the internal counter indicates a node has failed over, the server will no longer automatically failover additional nodes in the cluster.

You will need to re-enable automatic failover in a cluster by resetting this counter.

Avoiding failover chain-reactions (Cascading Failures)

Scenario:

- **Couchbase Server cluster of five nodes is operating at 80-90% aggregate capacity for network load**
- **A node crashes and auto-failover is triggered**
- **Increased load on remaining 4 nodes can cause a cascading failover and cause a cluster-wide outage!**
- **Solution: After a node crash, just continue cluster operations with 1/5th of the requests not being serviced due to node crash until an admin can replace the node with a healthy one**

45

Clearly having 1/5th of the requests not being serviced due to single node failure would be more desirable than none of the requests being serviced due to an entire cluster failure.

Solution: This way there is a brief partial outage rather than an entire cluster being disabled.

Good architectural design will have sized for this scenario and avoided it.

Handling failovers with network partitions

In case of network partition or split-brain where the failure of a network device causes a network to be split:

- Automatic failover requires a minimum of three (3) nodes per cluster.
- Automatic failover occurs only if exactly one (1) node is down.
- Automatic failover occurs only once before requiring administrative action.
- Automatic failover implements a 120 second delay when a node fails before it performs an automatic failover.

Rebalance after failover

After a node has been failed over, you should perform a rebalance operation. The rebalance operation will:

- **Redistribute stored data across the remaining nodes within the cluster.**
- **Recreate replicated data for all buckets at remaining nodes.**
- **Return your cluster to the configured operational state.**

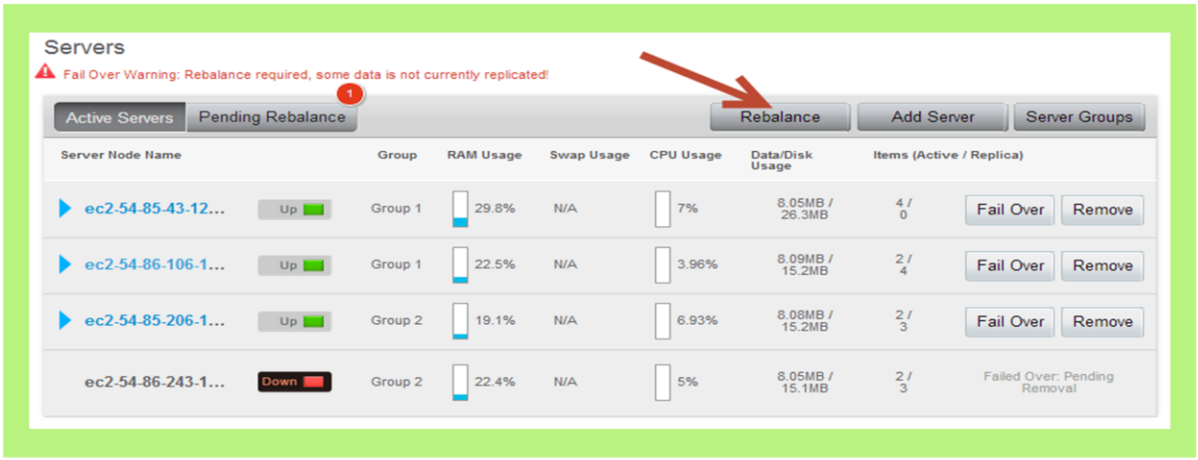
47

Adding back a failed over node

- Even after a node is failed over, the data on the failed node is intact, but the node will not be part of the cluster.
- When you add a failed over node back into a cluster, the cluster will treat it as if it is a new node(Full Recovery) or as a rejoining node containing possibly stale data(Delta Recovery)
- So you should rebalance after you add the node to the cluster. This rebalance will destroy any old data on the node!
- Before you add a failed over node back to the cluster, it is best practice to move or delete the persisted data files before you add the node back into the cluster.

Rebalance

Rebalance Concepts



50

Rebalance Concepts

Rebalance is the 2nd step after adding or removing nodes from a cluster

The addition and removal process merely configures a new node into the cluster, or marks a node for removal from the cluster. No actual changes are made to the cluster or data when configuring new nodes or removing existing ones.

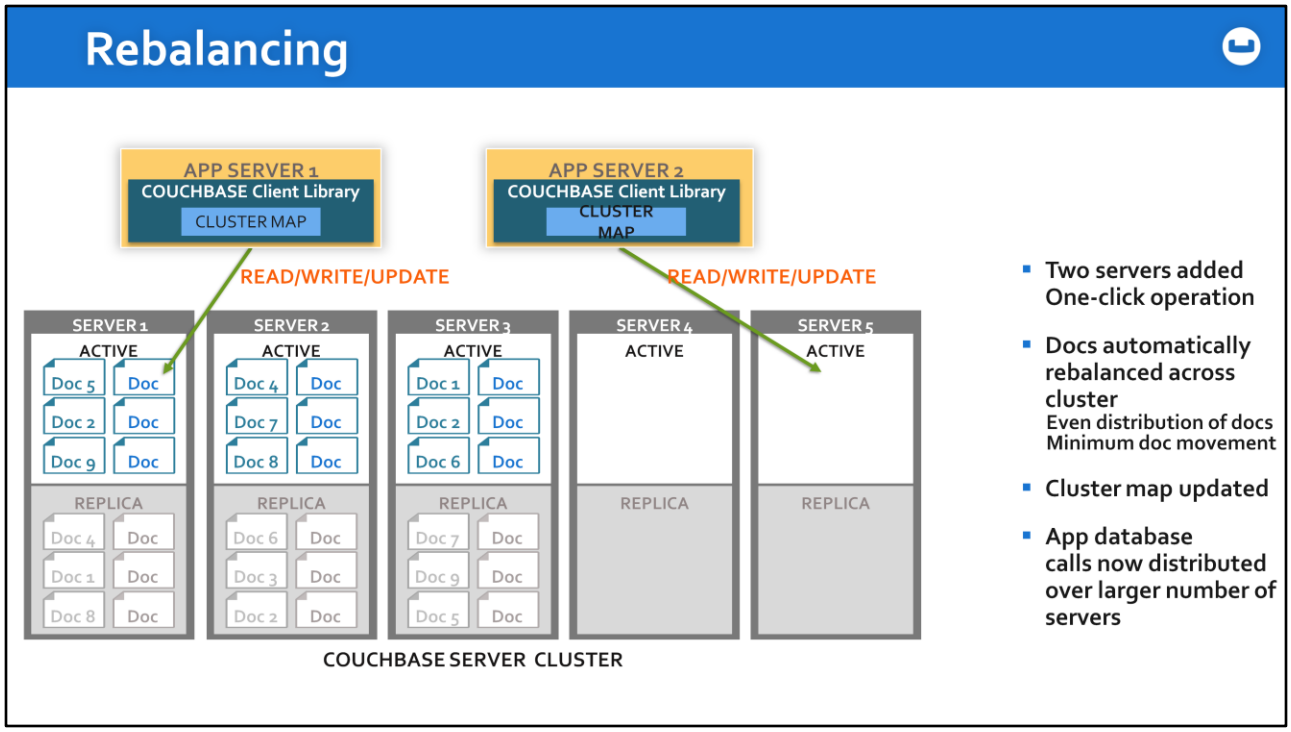
During Rebalance

The cluster remains up, and continues to service and handle client requests.

The current vBucket map is updated incrementally as each vBucket is moved.

Rebalancing may move both the data stored in RAM, and the data stored on disk for each bucket, and for each node.

The updated vBucket map is communicated to Couchbase client libraries and enabled smart clients and allows for continuous i/o during rebalance.



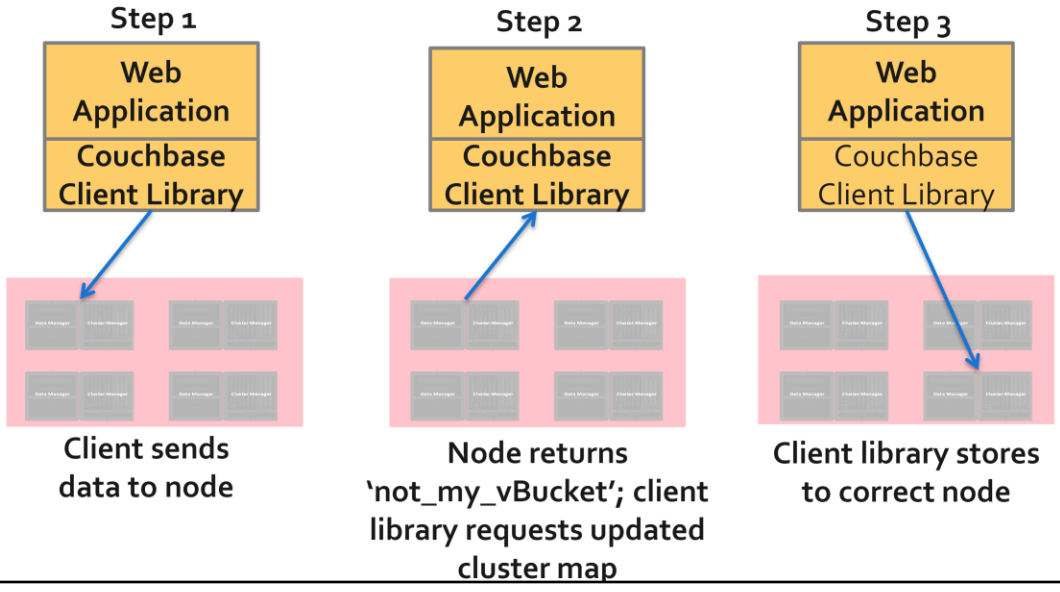
Now lets look at what happens when it comes time to add servers to the cluster. Starting with the same set of three nodes, we bring two more online (click). Note that you can add or remove multiple nodes at once before actually migrating any data. This helps greatly when needing to add or swap lots of nodes since you don't have to move the data around multiple times.

Once the administrator is ready, pressing the rebalance button (click) moves some of the active data and some of the replica data to the new nodes. Despite what the animation shows, this is actually done incrementally one shard (or vbucket) at a time which not only means that load is immediately and incrementally transferred to the new nodes, but this process can be stopped at any point in the middle and leave the cluster still in a stable, albeit unbalanced state.

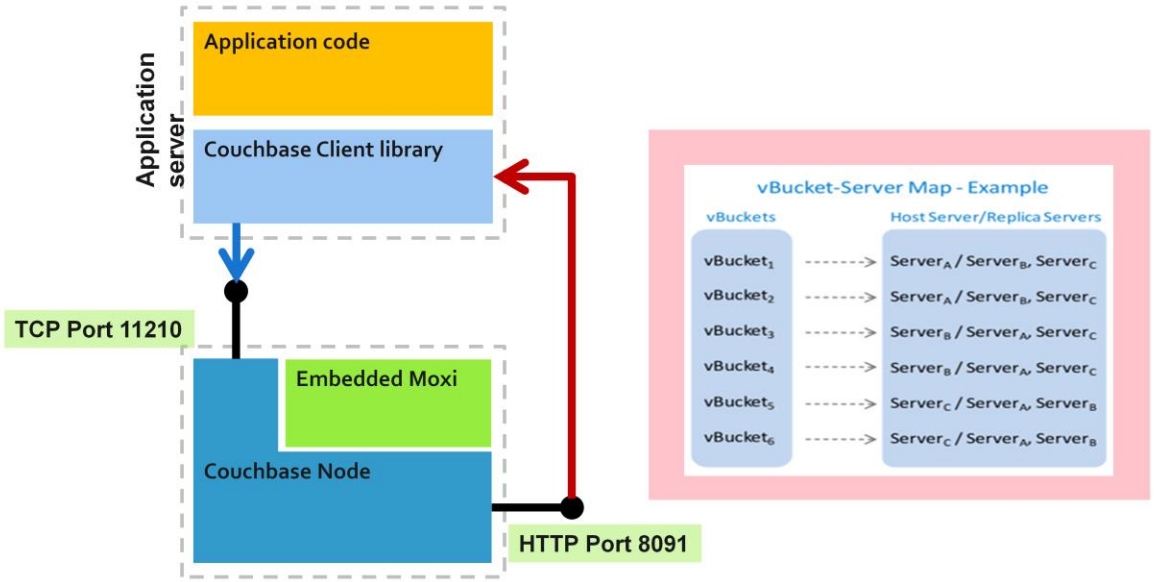
This whole process is done online which the application is accessing data. There is an atomic switchover for each shard as it is moved, and the application continues reading and writing data from the original location until that happens. Any writes are synchronized to the new location before switching over, and it is also replicated (and optionally persisted) to ensure data safety.

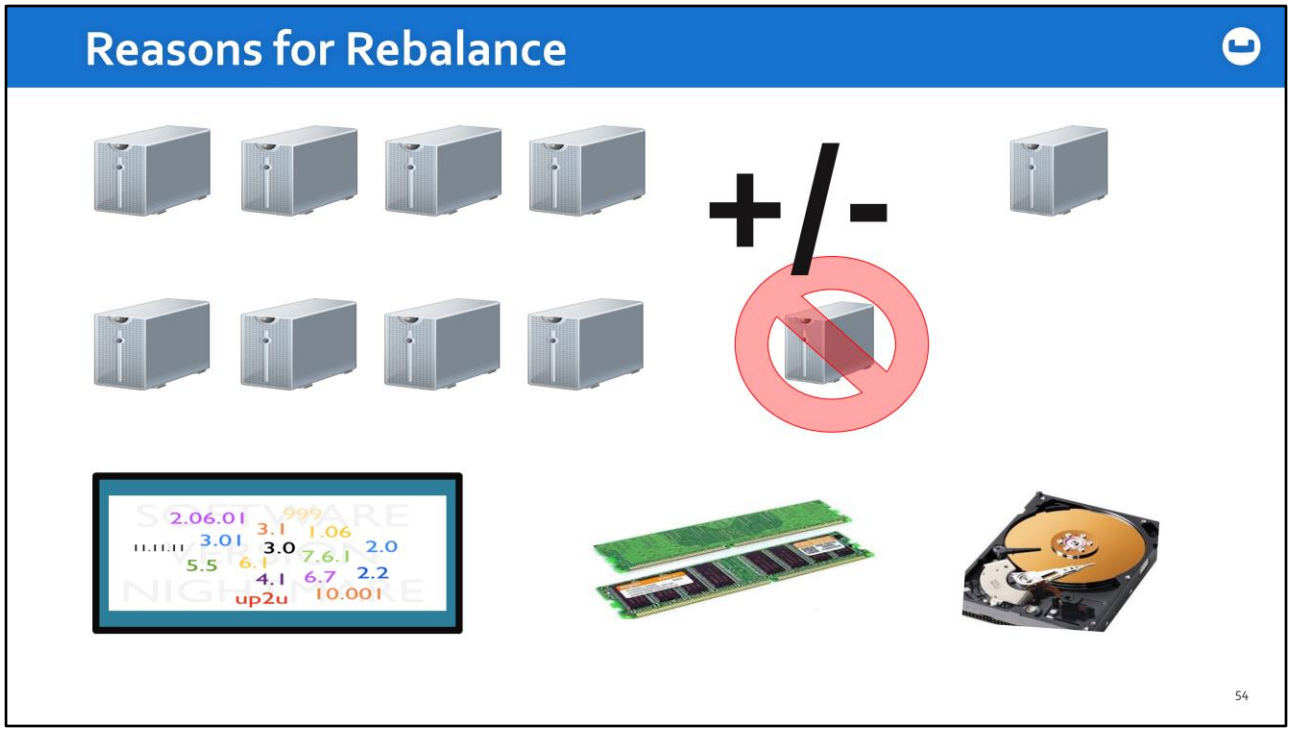
This same process can be used for software upgrades, hardware refreshes, and removing or swapping out misbehaving nodes.

Client Behavior during Rebalance



Cluster pushes new vbucket map when topology changes





Regardless of the reason for the rebalance, the purpose of the rebalance is migrate the cluster to a healthy state, where the configured nodes, buckets, and replicas match the current state of the cluster.
There are four primary reasons that you perform a rebalance operation:

Adding nodes to expand the size of the cluster.

Removing nodes to reduce the size of the cluster.

Reacting to a failover situation, where you need to bring the cluster back to a healthy state.

You need to temporarily remove one or more nodes to perform a software, operating system or hardware upgrade.

Best Practices

- When adding or removing multiple nodes, in a short period of time, it is best to add them all at once and then kick-off the rebalancing operation rather than rebalance after each addition.

- Choose a quiet time for adding nodes. While the rebalancing operation is meant to be performed online, it is not a “free” operation and will put increased load on the system as a whole in the form of disk IO, network bandwidth, CPU and RAM usage.

- Voluntary rebalancing (i.e. not part of a failover situation) should be performed during a period of low usage of the system.

- Rebalancing requires moving large amounts of data around the cluster. The more RAM that is available will allow the operating system to cache more disk access which will allow it to perform the rebalancing operation much faster.

55

Rebalancing via couchbase cli

```
#couchbase-cli rebalance -c 10.20.31.5:8091 -u
Administrator -p Password

INFO: rebalancing … … … … … … … … … … .
 … … … … … … … … … … … … … … .
 … … … … … … … … … … … … … … .
… … … … … … … … … … … … … … .
 … … … . .
SUCCESS: rebalanced cluster
```

56

Since the system receives messaging during rebalance it can be scripted and parsed.

Adding/Removing + Rebalancing via couchbase cli

```
# couchbase-cli rebalance -c 10.20.31.5:8091 \
            -u Administrator -p Password \
            -server-add=192.168.0.72 \
            -server-add=192.168.0.73 \
            -server-remove=192.168.0.70 \
            -server-remove=192.168.0.69
```

Note: You can also run

```
# couchbase-cli stop-rebalance
```

to stop a rebalance

57

Will this be a balanced swap ?
Or unbalanced swap?

Swap Rebalance

- Used for adding and removing the same number of nodes within the same operation

- Swap rebalance optimizes the rebalance operation by moving data directly from the nodes being removed to the nodes being added. (this is more efficient than normal rebalancing and moving data across entire cluster)

Swap rebalance only occurs if:

1. Removing and adding the same number of nodes during rebalance.
2. There are an even number of nodes being removed and added to the cluster

There is no configuration or selection mechanism to force a swap rebalance. If a swap rebalance cannot take place, then a normal rebalance operation will be used instead.

Swap Rebalance

During swap rebalance:

- Data will be moved directly from a node being removed to a node being added on a one-to-one basis. This eliminates the need to restructure the entire vBucket map.
- Active vBuckets are moved, one at a time, from a source node to a destination node.
- Replica vBuckets are created on the new node and populated with existing data before being activated as the live replica bucket. (so if there is a failure during the rebalance operation, the replicas are still in place)

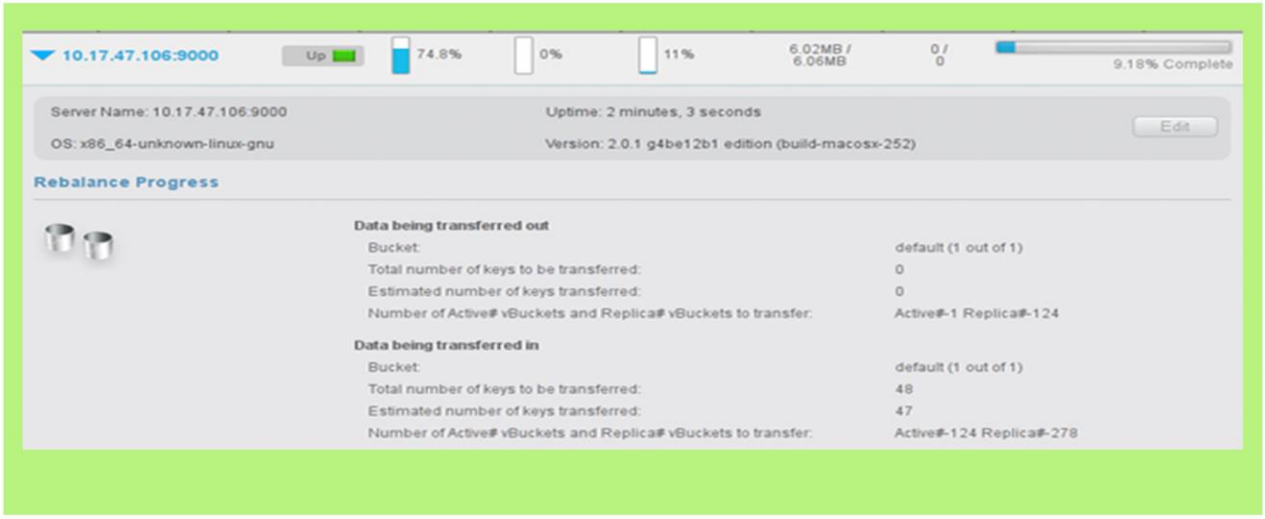
For example, if you have a cluster with 20 nodes in it, and configure two nodes (X and Y) to be added, and two nodes to be removed (A and B):

vBuckets from node A will be moved to node X.
vBuckets from node B will be moved to node Y.

Swap Rebalance Benefits

- Reduced rebalance duration. Since the move takes place directly from the nodes being removed to the nodes being added.
- Reduced load on the cluster during rebalance.
- Reduced network overhead during the rebalance.
- Reduced chance of a rebalance failure if a failover occurs during the rebalance operation, since replicas are created in tandem on the new hosts while the old host replicas still remain available.
- Because data on the nodes are swapped, rather than performing a full rebalance, the capacity of the cluster remains unchanged during the rebalance operation, helping to ensure performance and failover support.

Monitoring Rebalancing



61

Bucket : Name of bucket undergoing rebalance. Number of buckets transferred during rebalance out of total buckets in cluster.
Total number of keys : Total number of keys to be transferred during the rebalance.
Estimated number of keys : Number of keys transferred during rebalance.
Number of Active# vBuckets and Replica# vBuckets : Number of active vBuckets and replica vBuckets to be transferred as part of rebalance.

Backfilling metrics

When completed, you should see the Total Item count (curr_items_tot) be equal to the number of active items multiplied by replica count.

```
# cbstats <node_IP>:11210 -b bucket_name -p
bucket_password dcp | grep backfill
```

- After the backfill process is complete, all nodes that had replicas materialized on them will then need to persist those items to disk. It is important to continue monitoring the disk write queue and memory usage until the rebalancing operation has been completed, to ensure that your cluster is able to keep up with the write load and required disk I/O.

62

Rebalance behind-the-scenes

- Rebalance process is managed through a specific process called the orchestrator
- Orchestrator examines the current vBucket map and then combines that information with the node additions and removals in order to create a new vBucket map
- Orchestrator starts the process of moving the individual vBuckets from the current vBucket map to the new vBucket structure. The process is only started by the orchestrator - the nodes themselves are responsible for actually performing the movement of data between the nodes.
- Aim is to make the newly calculated vBucket map match the current situation
- On each destination node, a process called ebucketmigrator is started, which uses the DCP system to request that all the data is transferred for a single vBucket, and that the new vBucket data will become the active vBucket once the migration has been completed.
- Stopping rebalance will not reverse the status of already migrated buckets!

63

Changing the number of vBucket moves

- **Number of vBucket moves that occur during the rebalance operation can be modified.**
- **Default is 1**

To change the number of vBucket moves, execute a curl POST command using the following syntax with the /internalSettings endpoint and rebalanceMovesPerNode option:

```
# curl -X POST -u admin:password -d
rebalanceMovesPerNode=1 http://HOST:PORT/internalSettings
```

64

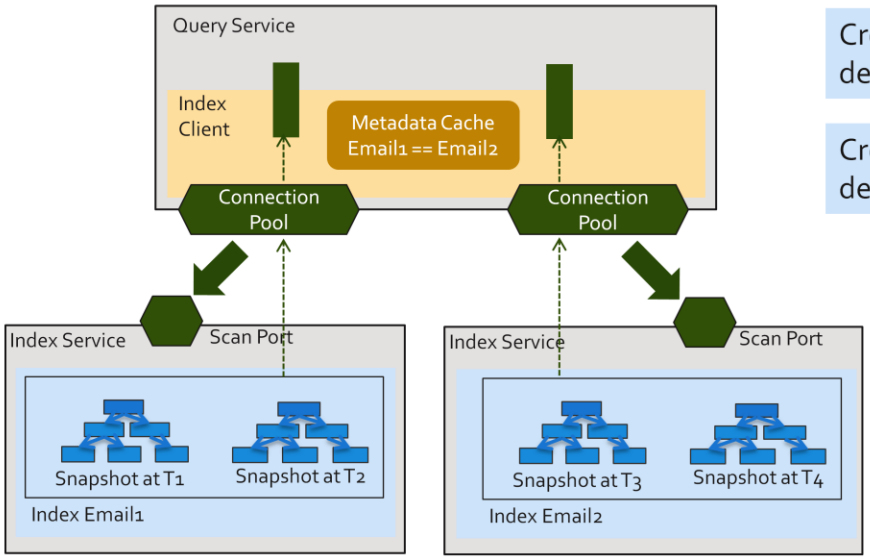
The default is one (1), that is, only one vBucket is moved at a time during the rebalance operation.

This REST command applies to the entire cluster.

This is a per-node setting.

Index Service
HA and Rebalance

Availability and Load Balancing



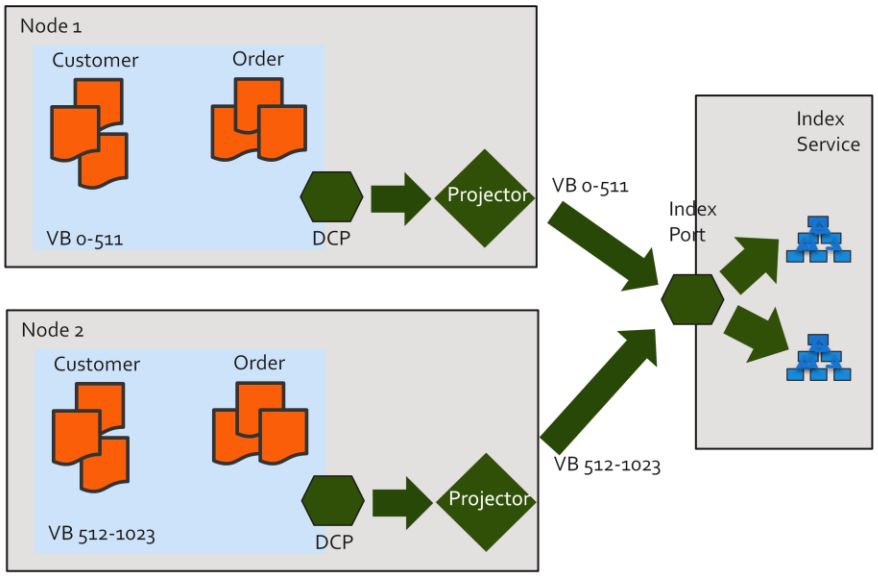
Create index Email1 on default(Email) using gsi;

Create index Email2 on default(Email) using gsi;

- Choose among equivalent indexes to serve scan request
- For loading balancing and read availability

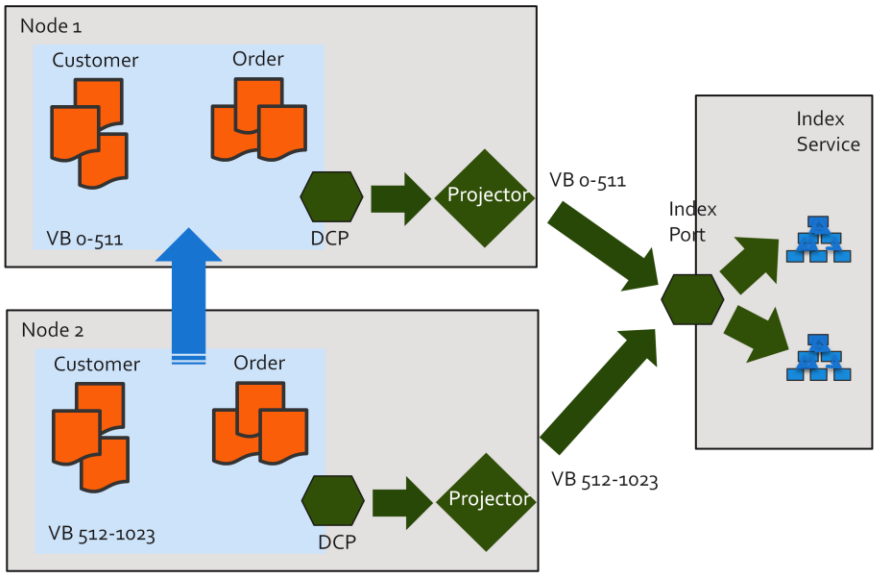
If two indexes have the same definitions, they can be used as read replica of each other.
In other words, they can be used for load balancing and read availability.

Data Rebalance



Now let's look into what happens to the index service when there is data rebalancing.
In this example, we have two data service nodes and a single index service node.
Let's say we are rebalancing out node 2.

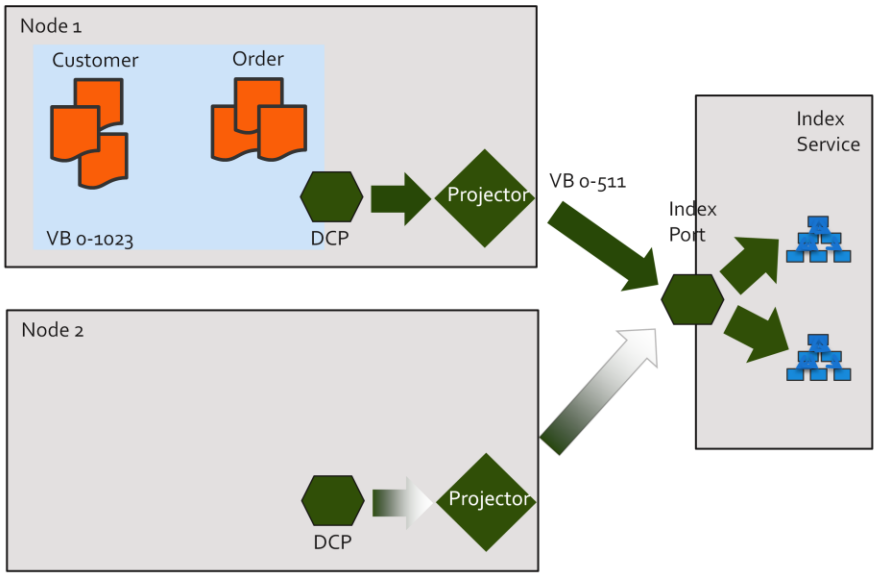
Data Rebalance



- Rebalance Vbucket 512-2013 from node 2 to node 1
- 2 Projectors continue to send keys to index service

While rebalancing is going on, both nodes will continue to send mutations to the index service.

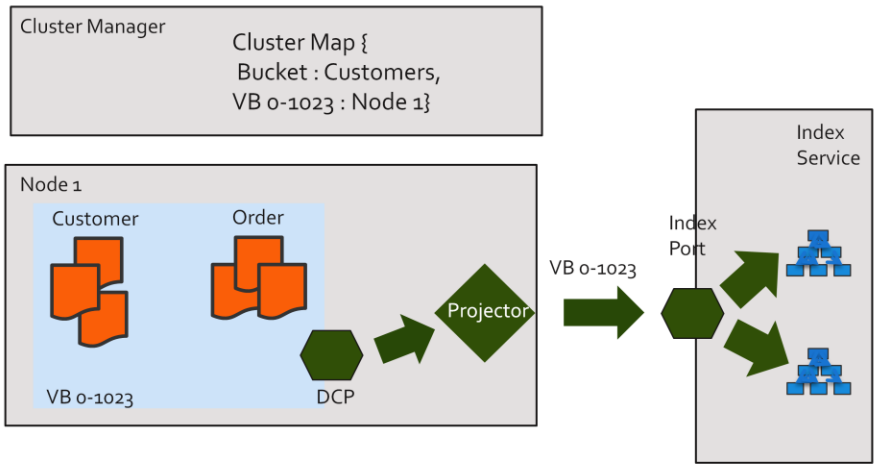
Data Rebalance



- Rebalance complete. Node 1 has Vbuckets 0-1023
- Node 1 projector continues to send mutations for 0-511
- Node 2 projector sends control messages for end of stream

As rebalance completes, the projector sends message to the index service to terminate the stream.

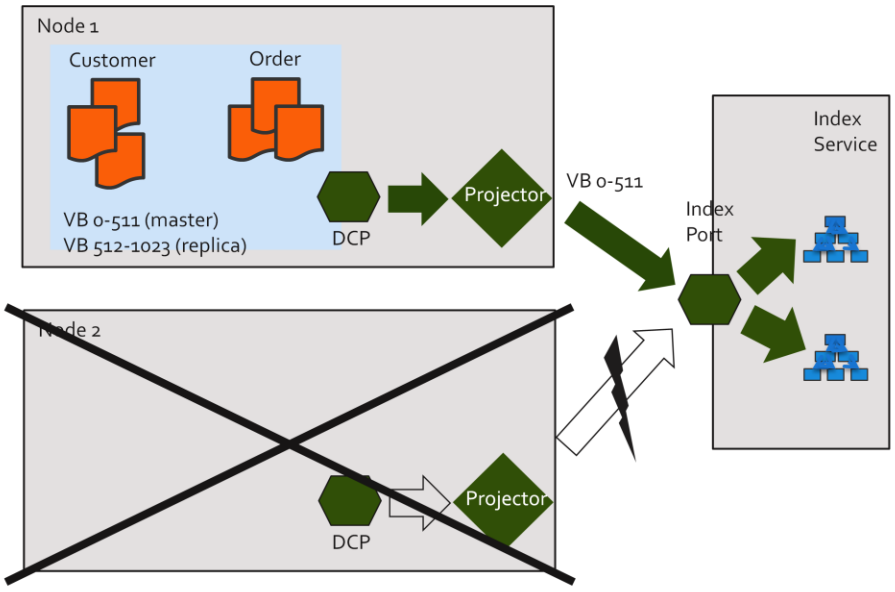
Data Rebalance



- Index Service receives end-of-stream for VB 512-1023
- Index Service finds out new VB master from cluster manager for VB 512-1023
- Index Service request Node 1 projector to include keys for VB 512-1023

The index service will then ask the cluster manager about the new master for the vbuckets.
It will then ask the projector from the new master to starting sending mtuations for the given vbuckets.

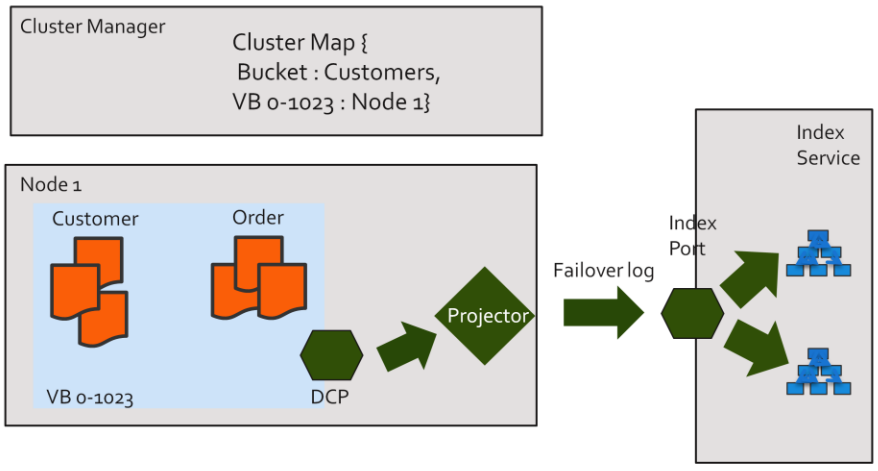
Data Failover



- Node 1 is the replica for VB 512-1023
- Node 2 fails
- Index Service receives connection error for Node 2

Handling data failover is similar to rebalancing.
In this case, the index service will receive a connection error.

Data Failover

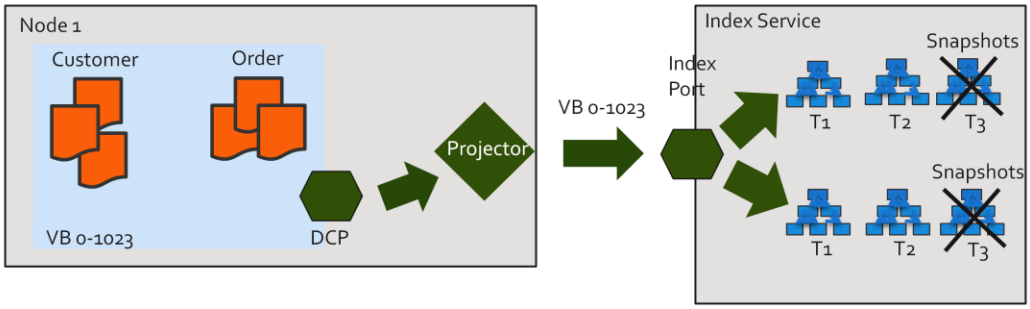


- Index Service finds out new VB master from cluster manager for VB 512-1023
- Index Service request Node 1 projector to send failover log

The index service will ask the cluster manager on the new vubcket master.
It will then ask for the failover log from the new master.

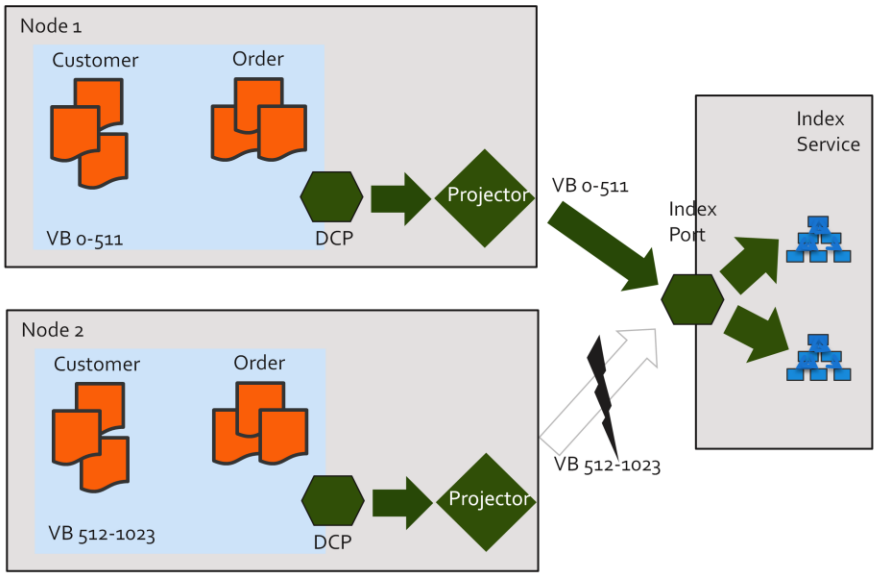
Data Failover

- From failover log, determine valid timestamp for rollback
- Valid timestamp = last valid sequence number of 1024 vbuckets
- Pick most recent persisted snapshot that matches rollback timestamp
- Rollback to snapshot
- Re-stream keys for vbuckets, starting from rollback snapshot



From the failover log, the indexer figures out the a valid timestamp for rollback.
A timestamp is the seqno of 1024 vbuckets.
The indexer then picks the most recent persisted snapshot that matches the rollback timestamp.
The data service will then re-stream any mutation starting from that timestamp.

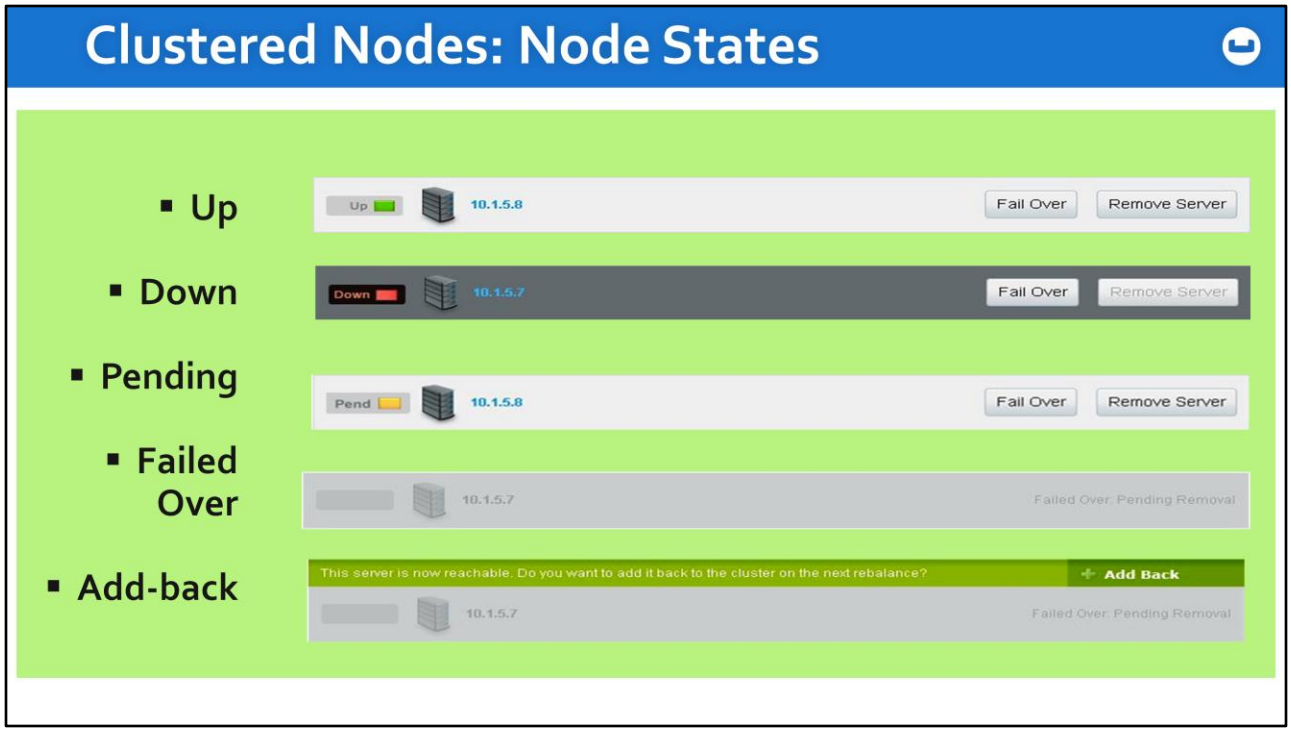
Network Partition



- Network partition is handled similarly as data failover, except
  - Snapshot rollback may not be required
  - Re-stream mutations from starting from when partition happens

Handling network partitioning is similar to data failover.
Except that the indexer does not have to rollback to a snapshot.

Cluster Node States



Understanding Cluster/Node States

Nodes can be in a variety of states to indicate:
Up (live and servicing client requests)
Down
Pending (cluster available, data not)
Failed over
Available to be added back
Pending Rebalance – action to be taken on next rebalance

Node status determined by individual bucket status per-node

Clustered Nodes: Node States

- Active Node List:

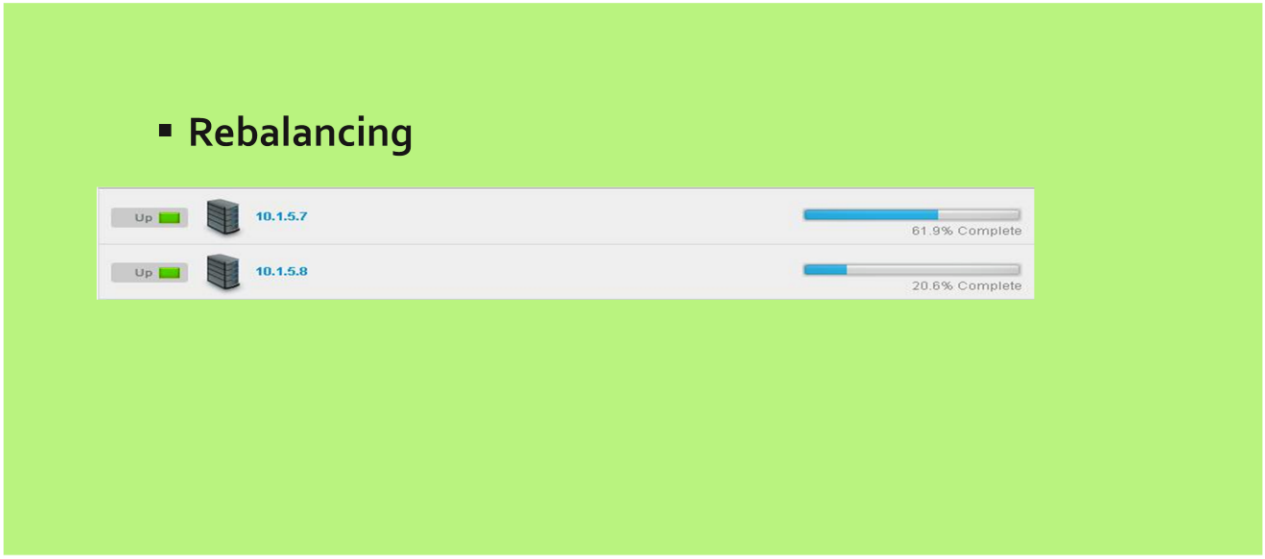
- Pending Rebalance List:

Understanding Cluster/Node States

Nodes can be in a variety of states to indicate:
Up (live and servicing client requests)
Down
Pending (cluster available, data not)
Failed over
Available to be added back
Pending Rebalance – action to be taken on next rebalance

Node status determined by individual bucket status per-node

Clustered Nodes: Node States



Understanding Cluster/Node States

Nodes can be in a variety of states to indicate:
Up (live and servicing client requests)
Down
Pending (cluster available, data not)
Failed over
Available to be added back
Pending Rebalance – action to be taken on next rebalance

Node status determined by individual bucket status per-node

Lab #4: Remove nodes & MISC commands

Time: 1 ½ hour

- More details on the Web UI's performance metrics
- Learn how to delete Couchbase buckets
- Gracefully decommission & recommission nodes from a cluster
- Failing over a node in the cluster
- Replica management and deeper understanding
- Using the REST API to check auto-failover settings

Couchbase

Learning Services