

CS300 Couchbase NoSQL Server Administration

Lab 4 Exercise Manual



Release: 6.5.1

Revised: June 22nd, 2020



Lab-4: Removing nodes, failover, misc, page 2

Lab #4: Gracefully removing nodes, replication, failover, MISC commands

Objective: This 90 min lab will cover some of the most important and common administrative features of couchbase, such as removing nodes, replication concepts, failover and rebalancing.

Overview: The following high-level steps are involved in this lab:

- Configure the linux .bash_profile for Couchbase commands
- Write 100,000 items to the 6-node Couchbase cluster using cbworkloadgen
- Change the workload balance between reads vs. writes in cbworkloadgen
- Increase the # of threads in cbworkloadgen
- Use the Web UI's performance metrics, to determine whether all nodes in the cluster are getting a balanced amount of i/o (with no skews towards certain nodes)
- Display individualized metrics for each node in the Web UI
- Delete Couchbase buckets
- Gracefully decommission and re-commission a node from the cluster
- Fail over a node in the cluster
- Replica management: Write 10 keys to Couchbase and track down which vBucket each key landed in using the 'cbc hash' command
- Use the REST API to check auto-failover settings and disable auto-failover

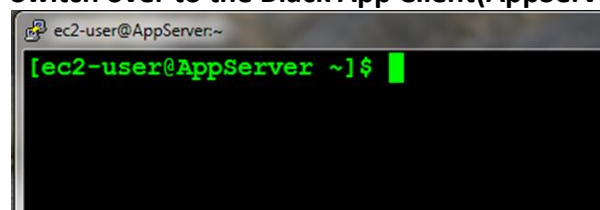


You are going to set Autofailover to off on your cluster(in a subsequent step)

Write data to the 6-node Couchbase Cluster and benchmark each node:

Now that the 6-node cluster is set up, let's try writing 100,000 items from the App Server to the 6-node cluster.

Switch over to the Black App Client(AppServer) PuTTY/Terminal shell



The next command will insert 100,000 items of size 10 bytes with 95% of the workload set to writes. This is the same test we did from our AppServer with just a 1-node Couchbase cluster, which resulted in about 151,400 operations per second (your mileage may have varied depending on cloud conditions).

In the next command, change the IP address to your 1st VM's public hostname:

```
[ec2-user@AppServer ~]$ cbworkloadgen -n $NODE1:8091 -i 100000 -r .95 -s 10
```

```
##### 100.0% (105264/estimated 105263 msgs)
bucket: default, msgs transferred...
      :                total |      last |      per sec
byte  :                1052640 |      1052640 |      202401.9
done
```

Notice that in the above output, the cluster is handling **202401.9 bytes** of I/O per second. Try running the same command with the '-v' option added to the end of the command for verbose mode:

```
[ec2-user@AppServer ~]$ cbworkloadgen -n $NODE1:8091 -i 100000 -r .95 -s 10 -v
2019-03-18 21:57:29,967: mt cbworkloadgen...
2019-03-18 21:57:29,967: mt source : gen=json=0,max-items=100000,low-
compression=False,prefix=pymc,ratio-sets=0.95,exit-after-creates=1,min-value-size=10
```



Lab-4: Removing nodes, failover, misc, page 4

```

2019-03-18 21:57:29,967: mt sink : http://ec2-13-56-188-91.us-west-
1.compute.amazonaws.com:8091
2019-03-18 21:57:29,967: mt opts : {'username': '<xxx>', 'destination_vbucket_state':
'active', 'verbose': 1, 'extra': {'max_retry': 10.0, 'rehash': 0.0, 'dcp_consumer_queue_length':
1000.0, 'data_only': 0.0, 'uncompress': 0.0, 'nmv_retry': 1.0, 'conflict_resolve': 1.0,
'cbb_max_mb': 100000.0, 'report': 5.0, 'mcd_compatible': 1.0, 'try_xwm': 1.0, 'backoff_cap': 0.1,
'batch_max_bytes': 400000.0, 'report_full': 2000.0, 'flow_control': 1.0, 'batch_max_size':
1000.0, 'seqno': 0.0, 'design_doc_only': 0.0, 'allow_recovery_vb_remap': 0.0, 'recv_min_bytes':
4096.0}, 'collection': None, 'ssl': False, 'threads': 1, 'key': None, 'password': '<xxx>', 'id':
None, 'destination_operation': None, 'source_vbucket_state': 'active', 'silent': False,
'dry_run': False, 'single_node': False, 'bucket_destination': 'default', 'vbucket_list': None,
'separator': '::', 'bucket_source': None}
2019-03-18 21:57:29,975: mt Starting new HTTP connection (1): ec2-13-56-188-91.us-west-
1.compute.amazonaws.com
2019-03-18 21:57:30,016: mt bucket: default
2019-03-18 21:57:35,029: w0 source : gen:json=0,max-items=100000,low-
compression=False,prefix=pymc,ratio-sets=0.95,exit-after-creates=1,min-value-size=10(default@N/A-
0)
2019-03-18 21:57:35,030: w0 sink : http://ec2-13-56-188-91.us-west-
1.compute.amazonaws.com:8091(default@N/A-0)
2019-03-18 21:57:35,030: w0 : total | last | per sec
2019-03-18 21:57:35,030: w0 batch : 106 | 106 | 21.1
2019-03-18 21:57:35,030: w0 byte : 1052640 | 1052640 | 209953.4
2019-03-18 21:57:35,030: w0 msg : 105264 | 105264 | 20995.3
[#####] 100.0% (105264/estimated 105263 msgs)
bucket: default, msgs transferred...
: total | last | per sec
batch : 106 | 106 | 20.3
byte : 1052640 | 1052640 | 201818.0
msg : 105264 | 105264 | 20181.8
done

```

You can see at the end of the above output, that the App Server was sending **20181 messages** per second into the cluster for a total of about **105263** messages.

We can also redo the test with purely 100% writes (notice that we changed the `-r` to 1):

```
[ec2-user@AppServer ~]$ cbworkloadgen -n ec2-54-85-43-x.compute-
1.amazonaws.com:8091 -u Administrator -p couchbase -i 100000 -r 1 -s
10 -v
```

<output truncated>

```

[#####] 100.0% (100000/estimated 100000 msgs)
bucket: default, msgs transferred...
: total | last | per sec
batch : 100 | 100 | 19.2
byte : 1000000 | 1000000 | 192240.4
msg : 100000 | 100000 | 19224.0
done

```

When running purely write workloads, the cluster is handling about **19224.0 write operations** per second.

Note : this will vary depending on number of Vcpu's

Try running the same command as above with an increased number of threads to 10 on the App Server to see if we can get better performance:

```
[ec2-user@AppServer ~]$ cbworkloadgen -n $NODE1:8091 -i 100000 -r 1 -s
10 -t 10 -v
```

```
2015-08-06 16:49:34,469: mt cbworkloadgen...
```



Lab-4: Removing nodes, failover, misc, page 5

<output truncated>

```
2015-08-06 16:50:14,478: w0 sink : http://ec2-52-6-74-39.compute-
1.amazonaws.com:8091 (default@N/A-1)
```

<output truncated>

```
2015-08-06 16:50:14,744: w2 source : gen:json=0,max-items=100000,low-
compression=False,prefix=pymc,ratio-sets=1.0,exit-after-creates=1,min-value-size=10 (default@N/A-
2)
```

```
2015-08-06 16:50:14,744: w2 sink : http://ec2-52-6-74-39.compute-
1.amazonaws.com:8091 (default@N/A-2)
```

```
2015-08-06 16:50:14,745: w2 : total | last | per sec
2015-08-06 16:50:14,746: w2 batch : 100 | 100 | 2.5
2015-08-06 16:50:14,746: w2 byte : 1000000 | 1000000 | 24863.1
2015-08-06 16:50:14,746: w2 msg : 100000 | 100000 | 2486.3
```

```
2015-08-06 16:50:14,780: w5 source : gen:json=0,max-items=100000,low-
compression=False,prefix=pymc,ratio-sets=1.0,exit-after-creates=1,min-value-size=10 (default@N/A-
6)
```

```
2015-08-06 16:50:14,781: w5 sink : http://ec2-52-6-74-39.compute-
1.amazonaws.com:8091 (default@N/A-6)
```

```
2015-08-06 16:50:14,781: w5 : total | last | per sec
2015-08-06 16:50:14,781: w5 batch : 100 | 100 | 2.5
2015-08-06 16:50:14,781: w5 byte : 1000000 | 1000000 | 24843.1
2015-08-06 16:50:14,782: w5 msg : 100000 | 100000 | 2484.3
```

```
2015-08-06 16:50:14,818: w4 source : gen:json=0,max-items=100000,low-
compression=False,prefix=pymc,ratio-sets=1.0,exit-after-creates=1,min-value-size=10 (default@N/A-
4)
```

<output truncated>

```
-size=10 (default@N/A-0)
```

```
2015-08-06 16:50:14,907: w1 sink : http://ec2-52-6-74-39.compute-
```

<output truncated>

```
2015-08-06 16:50:14,910: w9 source : gen:json=0,max-items=100000,low-
compression=False,prefix=pymc,ratio-sets=1.0,exit-after-creates=1,min-value-size=10 (default@N/A-
5)
```

```
2015-08-06 16:50:14,910: w9 sink : http://ec2-52-6-74-39.compute-
1.amazonaws.com:8091 (default@N/A-5)
```

```
2015-08-06 16:50:14,910: w9 : total | last | per sec
2015-08-06 16:50:14,910: w9 batch : 100 | 100 | 2.5
2015-08-06 16:50:14,911: w9 byte : 1000000 | 1000000 | 24763.8
2015-08-06 16:50:14,911: w9 msg : 100000 | 100000 | 2476.4
```

```
[#####] 99.9% (999000/estimated 1000000 msgs)
bucket: default, msgs transferred...
```

```
 : total | last | per sec
batch : 1000 | 1000 | 24.4
byte : 10000000 | 10000000 | 244138.6
msg : 1000000 | 1000000 | 24413.9
done
```

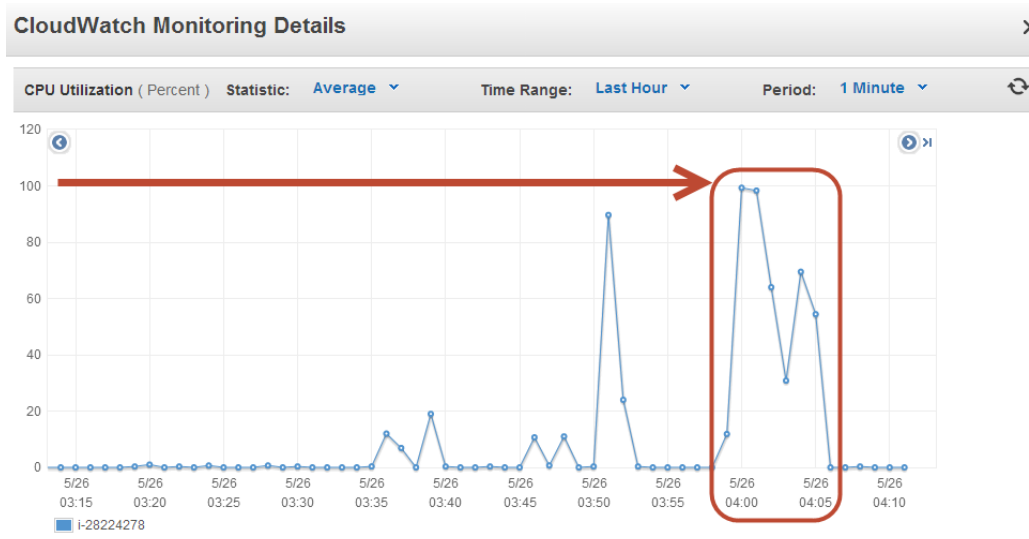
The above output will print out the statistics from each thread individually, so above you can see the **2nd (w2)** and **5th (w5)** threads.

However, notice that the overall write throughput was actually reduced down to **24413.9 per second**. Increasing the number of threads beyond a certain amount doesn't help performance, but actually hinders it.

The following screenshot was taken from Amazon CloudWatch dashboard and shows the CPU utilization of the App Server while the above command that generated 10 writer threads is running (see red circled area of the graph). *(Note that students will not have access to the CloudWatch metrics, however in the performance lab, different tools will be used to capture similar metrics.)*

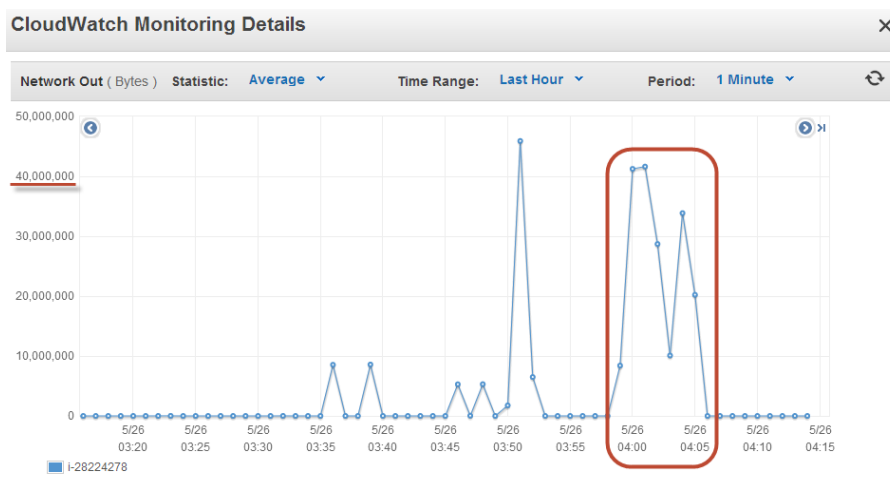


Lab-4: Removing nodes, failover, misc, page 6



Notice above that the CPU of the App Server is approaching 100% utilization when 10 threads are spawned.

The next screenshot from CloudWatch shows the Network out traffic in bytes on the App Server during the 10 thread write test. At its peak the App Server pushed 40,000,000 bytes or 305 Megabits of traffic (which is 38 Megabytes of traffic) out to the cluster.



Is there a way to find out how those roughly 13,000 - 14,000 writes per second are being distributed across the 4 nodes? It would be good to investigate whether any of the servers are getting a skewed amount of the incoming writes, so that some nodes are working harder than others.



Lab-4: Removing nodes, failover, misc, page 7

First, start a continuous stream of 100% writes from the App Server to the 6-node cluster. Use just 2 threads and write 10 byte sized items. The `-l` option in the command instructs the work load generator to loop forever until interrupted by the user:

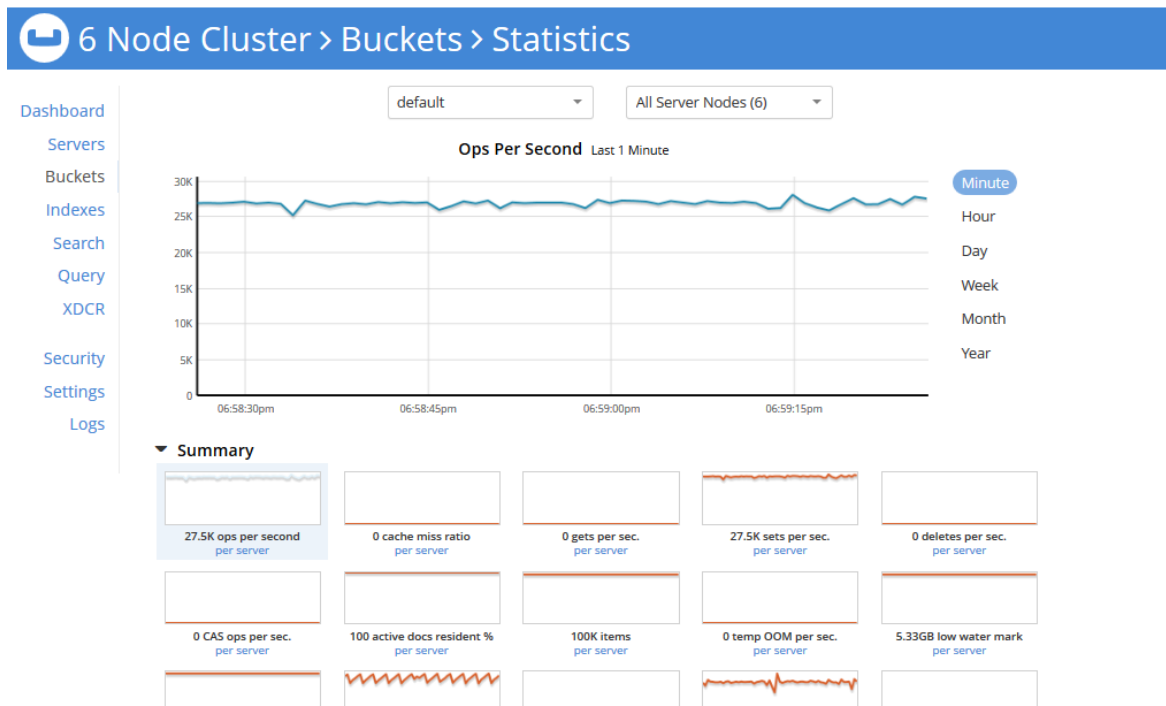
```
[ec2-user@AppServer ~]$ cbworkloadgen -n $NODE1:8091 -i 100000 -r 1 -s 10 -t 2 -v -l
```

While the above command is running in a loop, switch over to the Couchbase Web UI to look at some statistics to answer our question.

Click on **Buckets** link and the **statistics** link on the right hand side for the default bucket. to launch the graphs page:

6 Node Cluster > Buckets							ADD BUCKET
name	items	resident	ops/sec	RAM used/quota	disk used		
beer-sample	7,303	100%	0	58.3MB / 400MB	91.8MB	Documents Statistics	
default	100,000	100%	26685	44.4MB / 7.1GB	179MB	Documents Statistics	
gamesim-sample	586	100%	0	52.7MB / 400MB	64MB	Documents Statistics	

On the next statistics page, for the **default** data bucket on **All Server Nodes**:
You should be seeing approximately 20-30,000 operations per second:





Lab-4: Removing nodes, failover, misc, page 8

Scroll down and expand the SUMMARY section to notice the same 20-30,000 ops per second metric displayed again:

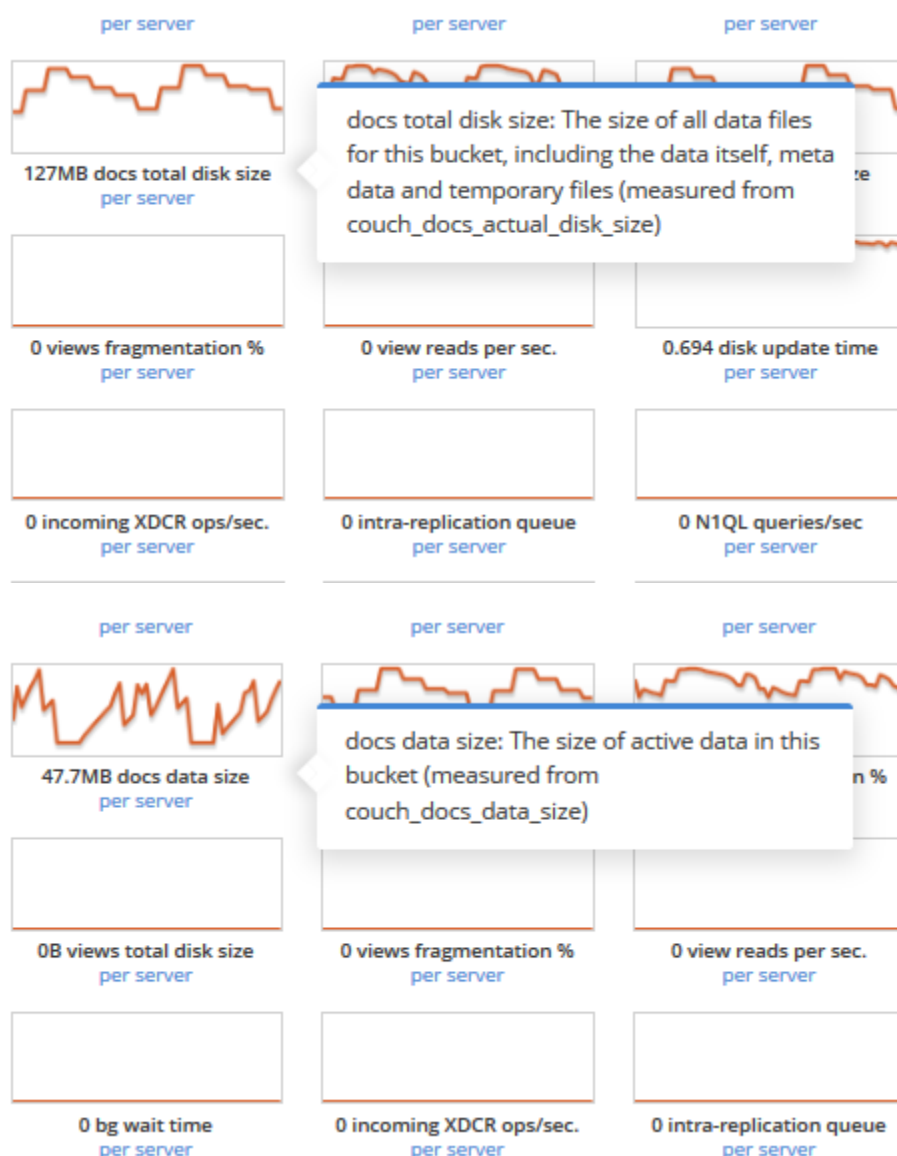
▼ Summary



In the SUMMARY section, notice that the total disk size chart and docs data size charts show how much space the default bucket is taking up on disk:
Your size may vary.



Lab-4: Removing nodes, failover, misc, page 9

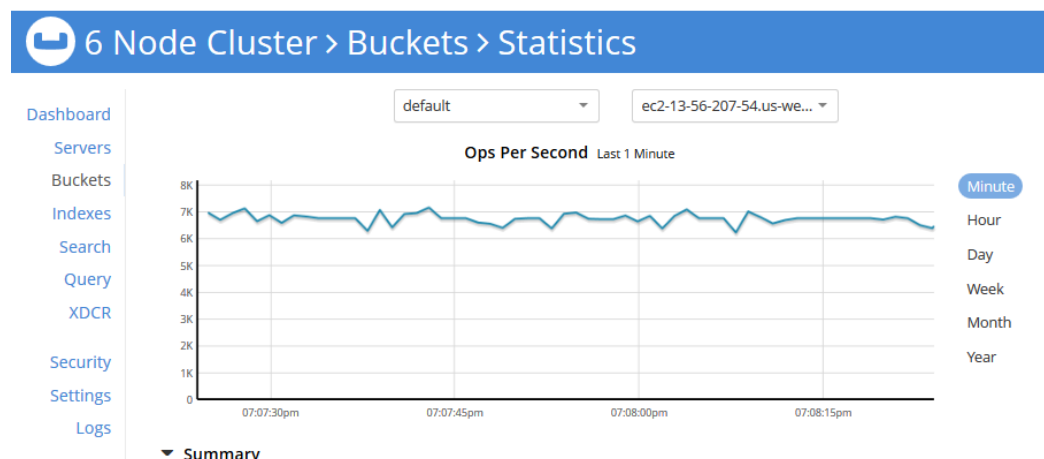


The total docs size is relatively low at 91 MB even though we are repeatedly inserting 30,000 items per second at a rate of about 40 MB per second. This is because the workload generator is re-writing 100,000 keys of size 10 bytes over and over again and not necessarily inserting fresh, new keys. Keep in mind that 10 bytes for each key multiplied by 100,000 is only 976 Kilobytes.

Scroll back up on this page and choose one of the 4 data service nodes in the cluster to display individualized metrics for:



Lab-4: Removing nodes, failover, misc, page 10

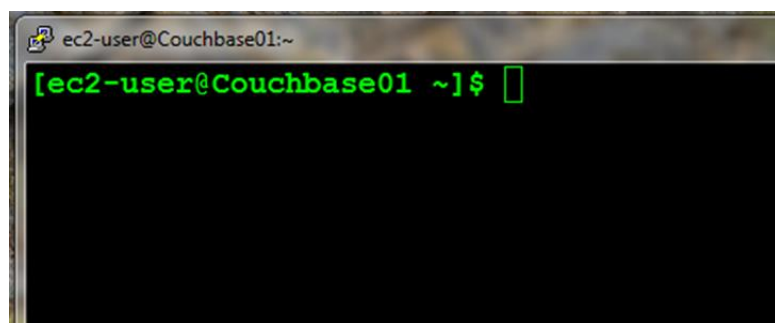


You should see roughly 6,000-8000 write operations being conducted on each of the 4 nodes. 6-8,000 ops per second multiplied by 4 nodes equals roughly 30,000 ops per second.

In the drop down, repeat the above step to verify that the other 3 nodes are also receiving about 6,000-8,000 ops per sec.

[Screenshots from the other 3 nodes not shown in lab]

Switch to the 1st node (Couchbase01) in the cluster and run the workload generator in a loop.



We will use the 1st Couchbase cluster node in a dual role to generate more writes for the cluster, and see how many more writes we can push into it. Perhaps the bottleneck currently is the App Server itself which cannot push any more than 30,000 items per second from its NIC in the cloud.

Run the workload generator from the 1st node with the exact same parameters as we ran it in a loop on the App Server:

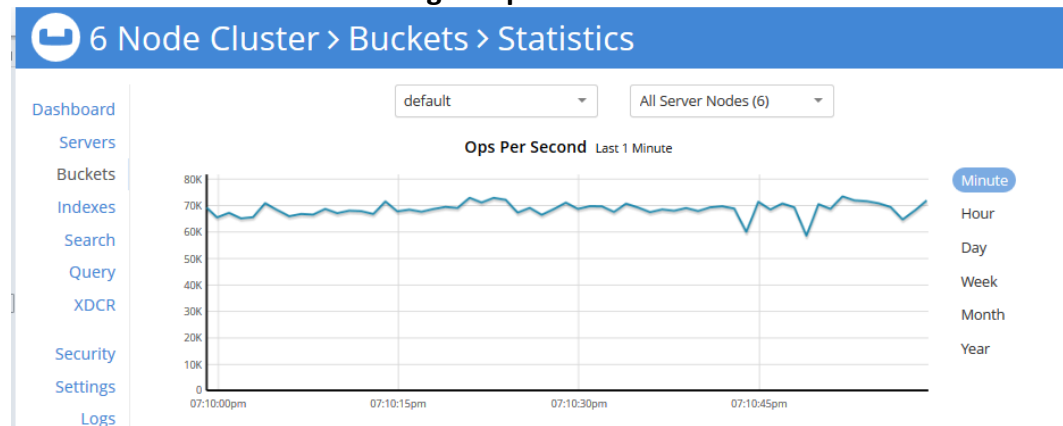


Lab-4: Removing nodes, failover, misc, page 11

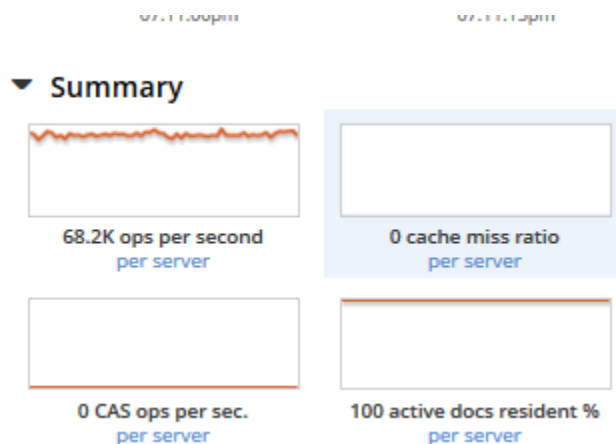
```
[ec2-user@Couchbase01 ~]$ cbworkloadgen -n $NODE1:8091 -i 100000 -r 1
-s 10 -t 2 -v -l
```

.....

From the Web UI, you should now see the cluster handling about 30,000 – 50,000 write operations per second across 4 data service nodes. You will also occasionally see a message that the 'default' bucket is being compacted:



Anyway, the SUMMARY section will show an exact count such as ~68.2 ops per second:

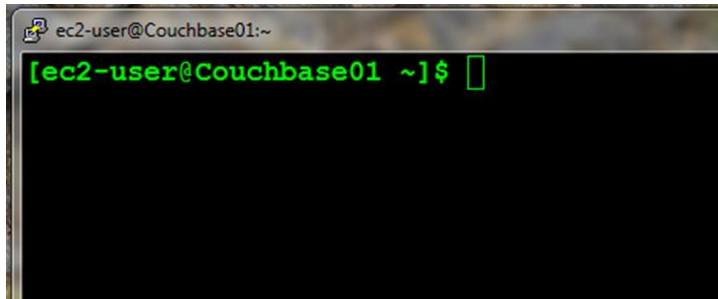


Using two machines to push the writes demonstrates that the App Server by itself is not capable of saturating the cluster with the amount of traffic that it can handle.



Lab-4: Removing nodes, failover, misc, page 12

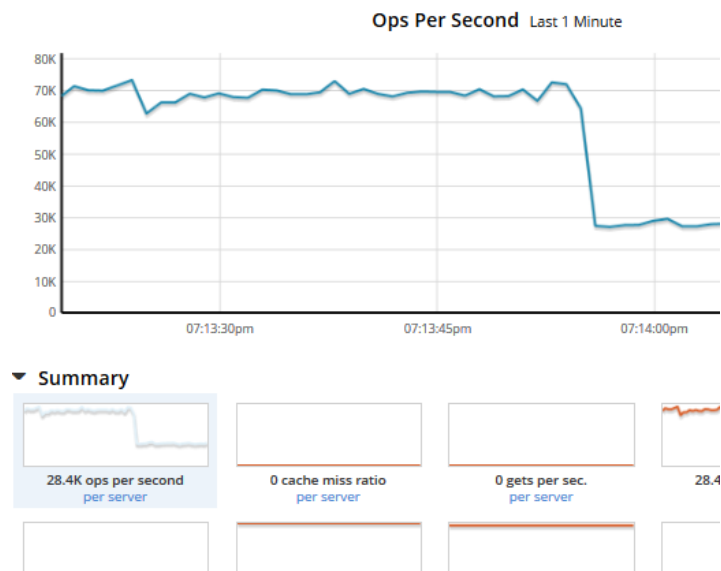
Return to the 1st Couchbase node (Couchbase01 dark blue VM) and **hit CTRL + C** to stop the **cbworkloadgen** command:



You will see a message such as this and be returned back to the command line:

```
.....^Cinterrupted.  
[ec2-user@Couchbase01 ~]$
```

In the Web UI you will see the cluster go back down to 15-20,000 write operations per second:





Gracefully decommission and recommission a node from the cluster:

Suppose you have to upgrade some hardware on a Couchbase cluster node and need to temporarily remove it from the cluster, power it down to upgrade the hardware and then start the server back up and re-join it to the cluster.

In scenarios like these, where you can plan your outage ahead of time, you should “remove”, “rebalance” and later on “add” the affected node back into the cluster.

In this section, we will walk through the correct method for removing a node from the cluster and adding the node back into the cluster. There is 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.

Switch over to the App Server (AppServer black VM) and stop the cbworkloadgenerator by **hitting CTRL + C**:



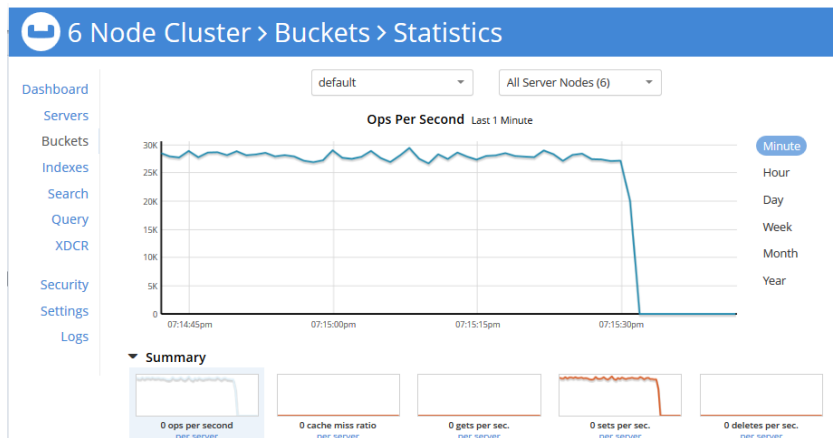
You will see a message such as this and be returned back to the command line:

```
.....^Cinterrupted.  
[ec2-user@AppServer ~]$
```

In the Web UI you will see the cluster go back down to 0 write operations per second:



Lab-4: Removing nodes, failover, misc, page 14



Switch to the Web UI and **click on “Buckets”** at the top and then **expand the beer-sample bucket**:

6 Node Cluster > Buckets

ADD BUCKET

name	items	resident	ops/sec	RAM used/quota	disk used	
beer-sample	7,303	100%	0	58.3MB / 400MB	91.8MB	Documents Statistics
<p>Type: Couchbase</p> <p>Bucket RAM Quota: 400MB</p> <p>Cluster RAM Quota: 2.07GB</p> <p>Replicas: 1</p> <p>Server Nodes: 4</p> <p>Ejection Method: Value-Only</p> <p>Conflict Resolution: Sequence Number</p> <p>Compaction: Not active</p>						
<p>Memory</p> <p>cluster quota (8.28 GiB)</p> <p>other buckets (7.5 GiB)</p> <p>this bucket (400 MB)</p> <p>remaining (400 MB)</p>						
<p>Disk</p> <p>total cluster storage (39.9 GiB)</p> <p>other buckets (101 MB)</p> <p>this bucket (91.8 MB)</p> <p>remaining (30.9 GiB)</p>						
default	100,000	100%	0	34.9MB / 7.1GB	37.6MB	Documents Statistics
gamesim-sample	586	100%	0	52.7MB / 400MB	64MB	Documents Statistics

Delete Compact Edit

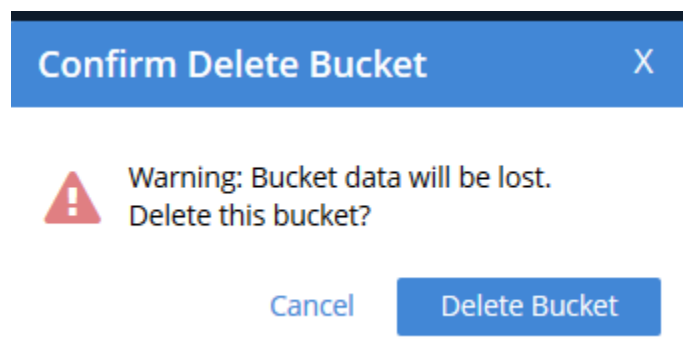
Once the details under beer-sample are expanded, **click on Delete**:

Delete

Click Delete on the Removing popup as well:



Lab-4: Removing nodes, failover, misc, page 15



In a few seconds, the bucket will be deleted:

Delete the gamesim-sample bucket next. **Expand the gamesim-sample details and then click "Delete":**

6 Node Cluster > Buckets

name	items	resident	ops/sec	RAM used/quota	disk used
default	100,000	100%	0	34.9MB / 7.1GB	37.6MB
gamesim-sample	586	100%	0	52.7MB / 400MB	64MB

gamesim-sample details:

- Type: Couchbase
- Bucket RAM Quota: 400MB
- Cluster RAM Quota: 2.07GB
- Replicas: 1
- Server Nodes: 4
- Ejection Method: Value Only
- Conflict Resolution: Sequence Number
- Compaction: Not active

Memory: cluster quota (8.28 GB), total cluster storage (39.9 GB)

Disk: other buckets (37.6 MB), this bucket (64 MB), remaining (31 GB)

Buttons: Delete, Compact, Edit

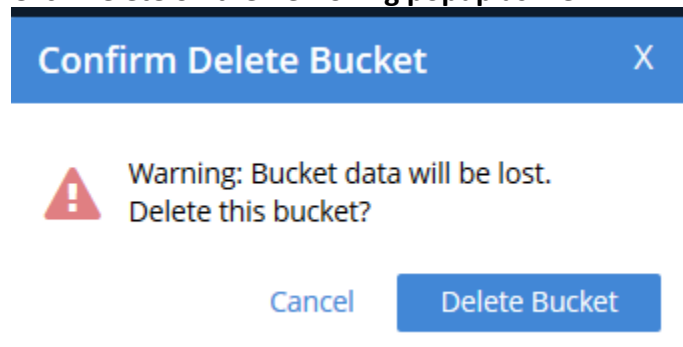
gamesim-sample summary:

- 4 replicas
- 586 items
- 0 resident
- 0 ops/sec
- 93.9MB / 400MB RAM
- 19.6MB / 30.8MB disk

Access Control: Authentication | Replicas: 1 replica copy | Compaction: Not active | Cache Metadata: Value Ejection | Disk I/O priority: Low

Buttons: Compact, Edit, Delete

Click Delete on the Removing popup as well:





Lab-4: Removing nodes, failover, misc, page 16

6 Node Cluster > Buckets							ADD BUCKET
name	items	resident	ops/sec	RAM used/quota	disk used		
default	100,000	100%	0	34.9MB / 7.1GB	37.6MB	Documents Statistics	

When completed will only have the default bucket left.

Finally, we will flush all of the contents in the default bucket. However, we will not delete the default bucket! **Expand the default bucket and click Flush:**

6 Node Cluster > Buckets							ADD BUCKET
name	items	resident	ops/sec	RAM used/quota	disk used		
default	100,000	100%	0	34.9MB / 7.1GB	37.6MB	Documents Statistics	
<div> <div>Type: Couchbase</div> <div>Bucket RAM Quota: 7.1GB</div> <div>Cluster RAM Quota: 2.07GB</div> <div>Replicas: disabled</div> <div>Server Nodes: 4</div> <div>Ejection Method: Value Only</div> <div>Conflict Resolution: Sequence Number</div> <div>Compaction: Not active</div> </div> <div> <div>Memory</div> <div>Cluster quota (8.28 GB)</div> <div>other buckets (0 B)</div> <div>this bucket (7.1 GB)</div> <div>remaining (1.17 GB)</div> </div> <div> <div>Disk</div> <div>total cluster storage (39.9 GB)</div> <div>other buckets (0 B)</div> <div>this bucket (37.6 MB)</div> <div>remaining (31.1 GB)</div> </div>							
							Delete Compact Flush Edit

Flush

NOTE: You have to first enable the Flush feature on the bucket(should have been done during setup in lab 2), before being able to flush it to empty it.
If not done, then click "edit" button>Advanced settings>
Scroll down and place a check next to Enable and then click Save:

Click Flush on the popup as well:

Confirm Flush Bucket

X

Warning: Bucket data will be lost.
Flush this bucket?

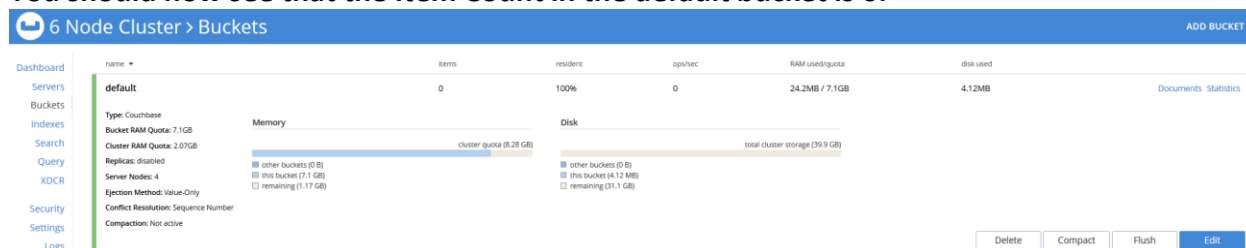
Cancel

Flush Bucket



Lab-4: Removing nodes, failover, misc, page 17

You should now see that the Item Count in the default bucket is 0:

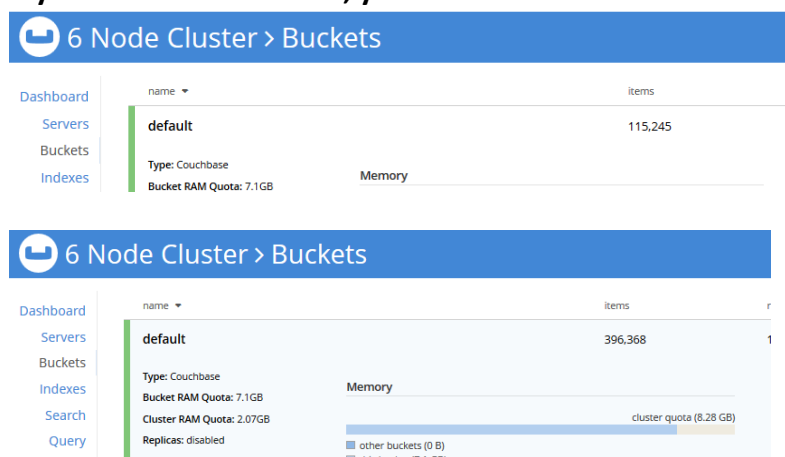


Switch over to the App Server (AppServer black VM) and generate a 100% write load against the default bucket. Use 1 million unique items, item size of 20 bytes, 2 threads and loop forever until interrupted. Note that this is about 150 MB of unique data.

```
[ec2-user@AppServer ~]$ while true; do cbworkloadgen -n $NODE1:8091 -i 1000000 -r 1 -s 20 -t 2 -v; sleep 2; done
```

NOTE: this will be in your command line history (up and down arrow keys) and can be edited for the correct item count

If you refresh the Web UI, you should see the Item Count for the default bucket increase:



Eventually, the Item Count will hit 1 million:



Lab-4: Removing nodes, failover, misc, page 18

6 Node Cluster > Buckets

Dashboard
Servers
Buckets
Indexes
Search
Query

name	items
default	1,000,000

Type: Couchbase
Bucket RAM Quota: 7.1 GB
Cluster RAM Quota: 2.07 GB
Replicas: disabled

Memory

cluster quota (8.28 GB)

other buckets (0 B)

Click on **“Servers”** at the right hand of the page and **identify the 4th machine** in the cluster. This should be in Group 2. Note, that the 4th machine will not necessarily be the last server in the list. Use the Cluster-IPs spreadsheet to match the public hostname of the 4th machine to one of the machines in this list. Then **click Remove for that machine**:

6 Node Cluster > Servers FILTER GROUPS ADD SERVER

Dashboard
Servers
Buckets
Indexes
Search
Query
XDCR
Security
Settings
Logs

name	group	services	CPU	RAM	swap	disk used	items	
ec2-13-56-207-54.us-west-1.compute.amazonaws.com	Group 2	data	32.3%	26.9%	---	80.6MB	250 K/O	Statistics
ec2-13-57-48-149.us-west-1.compute.amazonaws.com	Group 2	data	32.3%	19.9%	---	102MB	249 K/O	Statistics
Uptime: 4 days, 9 hours, 21 minutes OS: x86_64-unknown-linux-gnu Version: Enterprise Edition 5.0.0 build 3519 Data Service RAM Quota: 2.07 GB Data Storage Path: /opt/couchbase/var/lib/couchbase/data Index Storage Path: /opt/couchbase/var/lib/couchbase/index								
ec2-52-53-173-24.us-west-1.compute.amazonaws.com	Group 1	data	23.9%	20.2%	---	118MB	249 K/O	Statistics
ec2-54-183-85-83.us-west-1.compute.amazonaws.com	Group 1	data	24.4%	26.1%	---	34.7MB	250 K/O	Statistics
ec2-54-193-62-173.us-west-1.compute.amazonaws.com	Group 2	full text index query	1.01%	17.3%	---	---	0/0	Statistics
ec2-54-193-79-108.us-west-1.compute.amazonaws.com	Group 1	full text index query	1%	18.5%	---	---	0/0	Statistics

Remove Failover

Confirm the server removal by clicking **“Remove”**:

Confirm Server Removal X

Warning: Removing this server will...

- Reduce cache capacity across all data buckets

Cancel Remove Server

It will now appear with **“removal Pending Rebalance”**. The 4th node will technically not be removed from the cluster, until a rebalance.



Lab-4: Removing nodes, failover, misc, page 19

6 Node Cluster > Servers

Dashboard Servers Buckets Indexes Search Query XDCR Security Settings Logs

Rebalance

name	group	services	CPU	RAM	swap	disk used	items
ec2-13-56-207-54.us-west-1.compute.amazonaws.com	Group 2	data	52.8%	26.7%	---	29.9MB	250 K/O
ec2-13-57-48-149.us-west-1.compute.amazonaws.com	Group 2	data	33.5%	19.6%	---	66.2MB	249 K/O
Node flagged for removal Still taking traffic REMOVAL pending rebalance							
Uptime: 4 days, 9 hours, 23 minutes, 50 seconds OS: x86_64-unknown-linux-gnu Version: Enterprise Edition 5.0.0 build 3519 Data Service RAM Quota: 2.07 GB Data Storage Path: /opt/couchbase/var/lib/couchbase/data Index Storage Path: /opt/couchbase/var/lib/couchbase/index							
ec2-52-53-173-24.us-west-1.compute.amazonaws.com	Group 1	data	23.3%	20.5%	---	59.4MB	249 K/O
ec2-54-183-85-83.us-west-1.compute.amazonaws.com	Group 1	data	24.7%	25.7%	---	145MB	250 K/O
ec2-54-193-62-173.us-west-1.compute.amazonaws.com	Group 2	Full test Index Query	2.01%	17.6%	---	0/0	0/0
ec2-54-193-79-108.us-west-1.compute.amazonaws.com	Group 1	Full test Index Query	2.5%	18.6%	---	0/0	0/0

Click on Rebalance:

The Rebalance operation will start. Since there is only one bucket with around 150 MB of data in it, the rebalance should complete in under 2 minutes:

6 Node Cluster > Servers

Dashboard Servers Buckets Indexes Search Query XDCR Security Settings Logs

rebalancing 6 nodes 24.7%

Stop Rebalance

name	group	services	CPU	RAM	swap	disk used	items
ec2-13-56-207-54.us-west-1.compute.amazonaws.com	Group 2	data	26%	27.2%	---	107MB	250 K/O
ec2-13-57-48-149.us-west-1.compute.amazonaws.com	Group 2	data	35.2%	20.1%	---	146MB	249 K/O
Uptime: 4 days, 9 hours, 24 minutes, 35 seconds OS: x86_64-unknown-linux-gnu Version: Enterprise Edition 5.0.0 build 3519 Data Service RAM Quota: 2.07 GB Data Storage Path: /opt/couchbase/var/lib/couchbase/data Index Storage Path: /opt/couchbase/var/lib/couchbase/index Rebalance Progress Data being transferred out Total number of keys to be transferred: 249999 Estimated number of keys transferred: 92702 Number of active vBuckets and replica vBuckets to transfer: Active-161, Replica-0 Data being transferred in Total number of keys to be transferred: 0 Estimated number of keys transferred: 0 Number of active vBuckets and replica vBuckets to transfer: Active-0, Replica-0							
ec2-52-53-173-24.us-west-1.compute.amazonaws.com	Group 1	data	23.1%	20.4%	---	144MB	249 K/O
ec2-54-183-85-83.us-west-1.compute.amazonaws.com	Group 1	data	23.8%	25.8%	---	62.9MB	250 K/O
ec2-54-193-62-173.us-west-1.compute.amazonaws.com	Group 2	Full test Index Query	3.01%	17.3%	---	0/0	0/0
ec2-54-193-79-108.us-west-1.compute.amazonaws.com	Group 1	Full test Index Query	2%	18.6%	---	0/0	0/0

Switch to the App Server (black VM) and you will notice some warnings generated on the App Server such as:

```
2017-10-29 03:49:42,117: s0 warning: received NOT_MY_VBUCKET; perhaps the cluster is/was
rebalancing; vbucket_id: 342, key: pymc832873, spec: http://ec2-54-183-85-83.us-west-
1.compute.amazonaws.com:8091, host:port: ec2-13-57-48-149.us-west-1.compute.amazonaws.com:11210
2017-10-29 03:49:42,117: s2 warning: received NOT_MY_VBUCKET; perhaps the cluster is/was
rebalancing; vbucket_id: 439, key: pymc859387, spec: http://ec2-54-183-85-83.us-west-
1.compute.amazonaws.com:8091, host:port: ec2-13-57-48-149.us-west-1.compute.amazonaws.com:11210
```

After the rebalance is complete, the above warning messages will stop appearing.
When the rebalance is finished, you will see only 5 nodes displayed under Server Nodes.

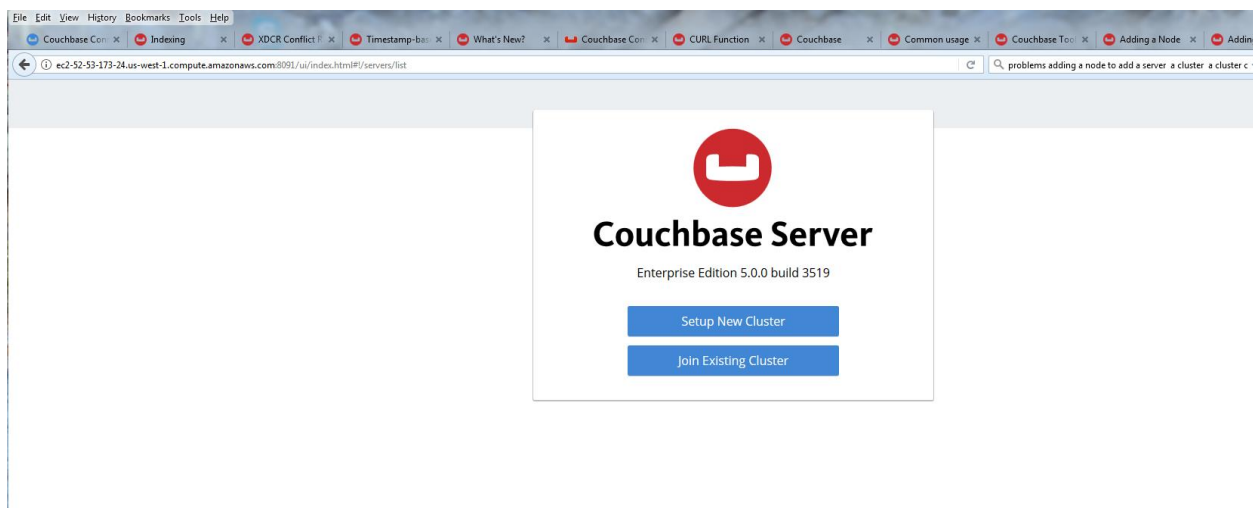


Lab-4: Removing nodes, failover, misc, page 20

6 Node Cluster > Servers									
FILTER GROUPS ADD SERVER									
Rebalance									
name	group	services	CPU	RAM	swap	disk used	items		
ec2-13-56-207-54.us-west-1.compute.amazonaws.com	Group 2	data	24%	27.3%	---	49.3MB	333 K/O	Statistics	
ec2-52-53-173-24.us-west-1.compute.amazonaws.com	Group 1	data	24.6%	20.3%	---	48.5MB	332 K/O	Statistics	
ec2-54-183-85-83.us-west-1.compute.amazonaws.com	Group 1	data	23.7%	25.5%	---	26.6MB	333 K/O	Statistics	
ec2-54-193-62-173.us-west-1.compute.amazonaws.com	Group 2	Full text Index Query	1%	17.2%	---	---	0/0	Statistics	
ec2-54-193-79-108.us-west-1.compute.amazonaws.com	Group 1	Full text Index Query	2.52%	18.7%	---	---	0/0	Statistics	

Next point your browser at **node 4** to obtain a setup splash screen

<http://ec2-52-53-173-24.us-west-1.compute.amazonaws.com:8091>



Choose

Join Existing Cluster

Fill in the information required making sure to use the Amazon ec2 address for both cluster and joining node name.

Select data service(deselect other services)



Lab-4: Removing nodes, failover, misc, page 21

Couchbase > Join Cluster

Cluster Host Name/IP Address

Cluster Admin Username

Cluster Admin Password

▼ Configure Services & Settings For This Node

☒ Data

☐ Index

☐ Search

☐ Query

☐ Eventing

☐ Analytics

This Node: Host Name/IP Address Usually localhost or similar

Data Disk Path Path cannot be changed after setup

Free: 6 GB

Indexes Disk Path Path cannot be changed after setup

Free: 6 GB

Analytics Disk Paths Paths cannot be changed after setup

Free: 6 GB + -

[< Back](#) [Join With Custom Configuration](#)

Edit index path to `/opt/couchbase/var/lib/couchbase/index`

Click

Join With Custom Configuration

Rebalancing the cluster:

On the servers tab click on the GROUPS link and move the new joining node to Group2



Lab-4: Removing nodes, failover, misc, page 22

6 Node Cluster > Servers > Server Groups

Dashboard Servers Buckets XDCR Security Settings Logs Documents Query Search Analytics Eventing Indexes

Apply Changes Add Group

Group 1 [edit name](#)

ec2-13-56-188-91.us-west-1.compute.a...	Data	move to ▼
ec2-13-56-209-124.us-west-1.compute....	Data	move to ▼
ec2-18-144-49-199.us-west-1.compute....	Data	move to ▼
ec2-54-183-155-27.us-west-1.compute....	Index Query Search	Group 2

Group 2 [edit name](#)

ec2-13-57-200-84.us-west-1.compute.a...	Data	move to ▼
ec2-54-183-183-230.us-west-1.comput...	Index Query Search	move to ▼

The UI should then look like this

6 Node Cluster > Servers > Server Groups

Dashboard Servers Buckets XDCR Security Settings Logs Documents Query Search Analytics Eventing Indexes

Reset Apply Changes Add Group

Group 1 [edit name](#)

ec2-13-56-188-91.us-west-1.compute.a...	Data	move to ▼
ec2-13-56-209-124.us-west-1.compute....	Data	move to ▼
ec2-18-144-49-199.us-west-1.compute....	Data	✓ pending move to Group 2 cancel
ec2-54-183-155-27.us-west-1.compute....	Index Query Search	move to ▼

Group 2 [edit name](#)

ec2-13-57-200-84.us-west-1.compute.a...	Data	move to ▼
ec2-54-183-183-230.us-west-1.comput...	Index Query Search	move to ▼

Click

Apply Changes

The UI should then appear as follows:



Lab-4: Removing nodes, failover, misc, page 23

6 Node Cluster > Servers > Server Groups

Dashboard Servers Buckets XDCR Security Settings Logs Documents Query Search Analytics Eventing Indexes

Apply Changes Add Group

Group 1 [edit name](#)

ec2-13-56-188-91.us-west-1.compute.a...	Data	move to ▼
ec2-13-56-209-124.us-west-1.compute....	Data	move to ▼
ec2-54-183-155-27.us-west-1.compute....	Index Query Search	move to ▼

Group 2 [edit name](#)

ec2-13-57-200-84.us-west-1.compute.a...	Data	move to ▼
ec2-18-144-49-199.us-west-1.compute....	Data	move to ▼
ec2-54-183-183-230.us-west-1.comput...	Index Query Search	move to ▼

Go back to the Servers link

You should see the 4th node in a Pending Add state. **Click on Rebalance:**

6 Node Cluster > Servers

FILTER GROUPS ADD SERVER

✓ This server has been associated with the cluster and will join on the next rebalance operation.

Rebalance

name	group	services	CPU	RAM	swap	disk used	items	
ec2-13-56-207-54.us-west-1.compute.amazonaws.com	Group 2	data	34%	27.2%	---	72MB	333 K/O	Statistics
ec2-13-57-48-149.us-west-1.compute.amazonaws.com	Group 1	data	1.49%	15.4%	---	---	0/0	Statistics
New node Not taking traffic ADD pending rebalance								
ec2-52-53-173-24.us-west-1.compute.amazonaws.com	Group 1	data	23.1%	20.7%	---	65.7MB	332 K/O	Statistics
ec2-54-183-85-83.us-west-1.compute.amazonaws.com	Group 1	data	25.4%	26.2%	---	148MB	333 K/O	Statistics
ec2-54-193-62-173.us-west-1.compute.amazonaws.com	Group 2	full text index query	1%	17.3%	---	---	0/0	Statistics
ec2-54-193-79-108.us-west-1.compute.amazonaws.com	Group 1	full text index query	1%	18.3%	---	---	0/0	Statistics

Cancel Add

Within 2 minutes, the rebalance operation should finish and the 4th node will be successfully added back into the cluster.

6 Node Cluster > Servers

rebalancing 6 nodes 16.7%

Stop Rebalance

name	group	services	CPU	RAM	swap	disk used	items	
ec2-13-56-207-54.us-west-1.compute.amazonaws.com	Group 2	data	48.7%	26.3%	---	107MB	333 K/O	29.1% complete Statistics
ec2-13-57-48-149.us-west-1.compute.amazonaws.com	Group 1	data	1%	14.3%	---	---	0/0	25% complete Statistics
ec2-52-53-173-24.us-west-1.compute.amazonaws.com	Group 1	data	36.8%	19.8%	---	141MB	332 K/O	22.4% complete Statistics
ec2-54-183-85-83.us-west-1.compute.amazonaws.com	Group 1	data	25.3%	24.9%	---	88.4MB	333 K/O	23.5% complete Statistics
ec2-54-193-62-173.us-west-1.compute.amazonaws.com	Group 2	full text index query	1.5%	17%	---	---	0/0	0% complete Statistics
ec2-54-193-79-108.us-west-1.compute.amazonaws.com	Group 1	full text index query	1.5%	17.9%	---	---	0/0	0% complete Statistics



Lab-4: Removing nodes, failover, misc, page 24

During this last rebalance, you will notice that the cbworkloadgen application on the App Server will continue to run:

```
2017-10-29 04:02:47,923: s0 warning: received NOT_MY_VBUCKET; perhaps the cluster is/was
rebalancing; vbucket_id: 683, key: pymc482114, spec: http://ec2-54-183-85-83.us-west-
1.compute.amazonaws.com:8091, host:port: ec2-52-53-173-24.us-west-1.compute.amazonaws.com:11210
2017-10-29 04:02:47,923: s0 warning: received NOT_MY_VBUCKET; perhaps the cluster is/was
rebalancing; vbucket_id: 683, key: pymc482666, spec: http://ec2-54-183-85-83.us-west-
1.compute.amazonaws.com:8091, host:port: ec2-52-53-173-24.us-west-1.compute.amazonaws.com:11210
2017-10-29 04:02:47,934: s0 refreshing sink map: http://ec2-54-183-85-83.us-west-
1.compute.amazonaws.com:8091
2017-10-29 04:02:47,938: s0 Starting new HTTP connection (1): ec2-54-183-85-83.us-west-
1.compute.amazonaws.com
2017-10-29 04:02:48,001: s1 warning: received NOT_MY_VBUCKET; perhaps the cluster is/was
rebalancing; vbucket_id: 683, key: pymc492819, spec: http://ec2-54-183-85-83.us-west-
1.compute.amazonaws.com:8091, host:port: ec2-52-53-173-24.us-west-
1.compute.amazonaws.com:11210error: MCSink exception:
```

To stop the loop in your appserver window do the following:

```
[ec2-user@appserver ~]$ [Cntrl] + [z]          (keys to background process)
```

```
[ec2-user@appserver ~]$ ps -elf (to identify cbworkloadgen process id)
```

```
UID          PID      PPID
```

```
ec2-user    24193   22920
```

```
[ec2-user@appserver ~]$ kill -9 24193
```

Failing over a node in the cluster:

If a node in a cluster is unable to serve data you can failover that node. Failover means that Couchbase Server removes the node from a cluster and makes replicated data at other nodes available for client requests. As long as you have a sufficient number of replicas configured in Couchbase, the cluster can handle failure of one or more nodes without affecting your ability to access the stored data. Failover can be initiated manually via the Web UI, REST API or Couchbase CLI.

Alternatively, you can configure Couchbase server to automatically remove a failed node from a cluster and have the cluster operate in a degraded mode. If you choose this automatic option,



the workload for functioning nodes that remain the cluster will increase. You will still need to address the node failure, return a functioning node to the cluster and then rebalance the cluster in order for the cluster to function as it did prior to node failure. Automatically failing components in any distributed system can cause problems. If you cannot identify the cause of failure, and you do not understand the load that will be placed on the remaining system, then automated failover can cause more problems than it is designed to solve.

Keep in mind that if you try to failover a functioning node it may result in data loss. This is because failover will immediately remove the node from the cluster and any data that has not yet been replicated to other nodes may be permanently lost if it had not been persisted to disk.

Be aware that failover is a distinct operation compared to removing/rebalancing a node. Typically you remove a functioning node from a cluster for maintenance, or other reasons; in contrast you perform a failover for a node that does not function. If you need to remove a functioning node for administration purposes, you should use the remove and rebalance functionality not failover.

Before we trigger an actual Failover, let's understand the set up and configuration of the default bucket, where its vBuckets are stored and whether autoFailover is turned on for our lab cluster.

Run the next set of commands from the App server (Black VM):



Run the cbstats command against the 1st node's public hostname and grep for active_num and replica_num:

```
[ec2-user@AppServer ~]$ cbstats $NODE1:11210 all -u Administrator -p couchbase -b default | grep active_num
vb_active_num: 256
vb_active_num_non_resident: 0
[ec2-user@AppServer ~]$ cbstats $NODE1:11210 all -u Administrator -p couchbase -b default | grep replica_num
vb_replica_num: 0
vb_replica_num_non_resident: 0
```

Notice that the 1st node is only hosting 256 active vBuckets and no replica vBuckets.



Lab-4: Removing nodes, failover, misc, page 26

Each bucket in Couchbase is made of 1024 individual vBuckets (virtual buckets). The 1024 vBuckets are distributed throughout the cluster evenly so each node is hosting a somewhat fair load of the bucket.

Try running the same command against the 2nd node's public hostname (get this from the Cluster-IPs spreadsheet):

```
[ec2-user@AppServer ~]$ cbstats $NODE2:11210 all -u Administrator -p couchbase -b default | grep active_num
```

```
vb_active_num: 256
vb_active_num_non_resident: 0
```

```
[ec2-user@appserver ~]$ cbstats $NODE2:11210 all -u Administrator -p couchbase -b default | grep replica_num
```

```
vb_replica_num: 0
vb_replica_num_non_resident: 0
```

You should see 256 active-num on the 2nd node as well.

Note that since we don't have any replicas configured for the default bucket, we are not utilizing the Rack Awareness/Server Groups feature to distribute replica vBuckets to different Server Groups.

Let's switch to the Web UI and enable 1 replica for the default bucket. But first flush the default bucket to delete all of the data within it.

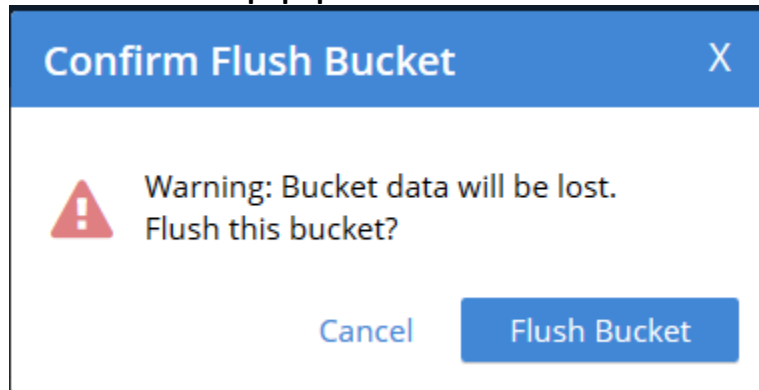
Click on **Buckets** at the left hand side of the Web UI, notice the **Item Count** in the bucket (your item count # may differ than the # in the screenshot below) and then click on **Flush**:

The screenshot shows the Couchbase Web UI interface. At the top, there's a blue header bar with the Couchbase logo and '6 Node Cluster > Buckets'. On the left, a sidebar contains navigation links: Dashboard, Servers, Buckets, Indexes, Search, Query, XDCR, Security, Settings, and Logs. The main content area displays the 'default' bucket. It shows a table with columns: name, items, resident, ops/sec, RAM used/quota, and disk used. The 'default' bucket has 1,000,000 items, is 100% resident, has 0 ops/sec, and uses 137MB of 7.1GB RAM. Below the table, there are two bar charts: 'Memory' and 'Disk'. The 'Memory' chart shows the bucket's RAM usage (7.1 GB) relative to the cluster quota (8.28 GB). The 'Disk' chart shows the bucket's disk usage (84.4 MB) relative to the total cluster storage (39.9 GB). At the bottom right, there are buttons for 'Delete', 'Compact', 'Flush', and 'Edit'.

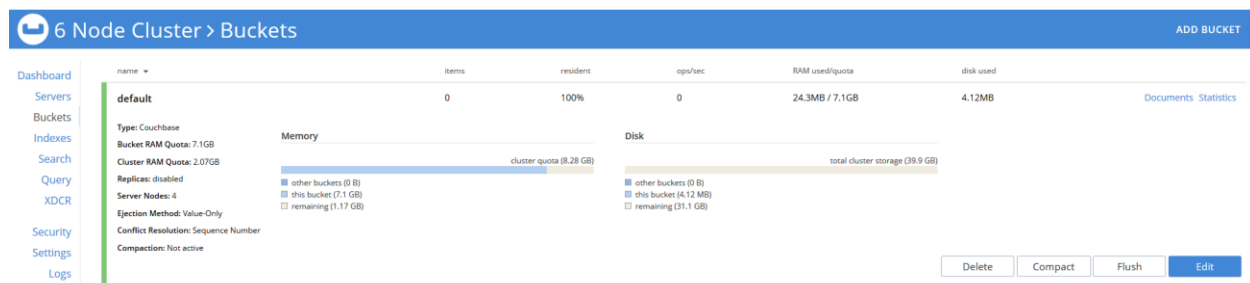


Lab-4: Removing nodes, failover, misc, page 27

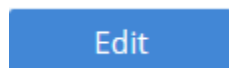
Click Flush on the popup as well:



You will now see that the item count is 0 and that Replicas are still disabled for this bucket:



Enable one replica for the default bucket next. **Click on Edit** again:



Expand “Advanced bucket settings” you should see a setting for Replicas. **Place a check mark next to Enable** and **ensure that the Number of replica copies is set to 1:**



Lab-4: Removing nodes, failover, misc, page 28

Edit Bucket Settings X

Name

Memory Quota in megabytes per server node
 MB

other buckets (0 B) this bucket (7.1 GB) remaining (1.17 GB)

Bucket Type
☒ Couchbase ☐ Memcached ☐ Ephemeral


▼ **Advanced bucket settings**

Replicas
☒ Enable Number of replica (backup) copies
Warning: changing replica number may require rebalance.
☐ Replicate view indexes

Conflict Resolution ⓘ
☒ Sequence number ☐ Timestamp

Ejection Method ⓘ
☒ Value-only ☐ Full

Replicas

 ☒ Enable Number of replica (backup) copies
Warning: changing replica number may require rebalance.
☐ Index replicas

Click Save:**Save Changes****You should now see 1 replica copy configured:**



6 Node Cluster > Buckets

Dashboard
Servers
Buckets
Indexes
Search
Query
XDCR
Security
Settings
Logs

name ▼

default

Type: Couchbase
Bucket RAM Quota: 7.1GB
Cluster RAM Quota: 2.07GB
Replicas: 1
Server Nodes: 4
Ejection Method: Value-Only
Conflict Resolution: Sequence Number
Compaction: Not active

Memory

other buckets (0 B)
this bucket (7.1 GB)
remaining (1.17 GB)

In the Couchbase Web UI, **click on “Servers”** page ,Notice that there are no specific nodes that need to be rebalanced at this time, even though we added a replica to the default bucket. This is because the default bucket is empty and contains zero keys since we flushed it earlier. However, also notice that there is a orange bar with a message **Warning: Rebalance required, some data is not currently replicated.**

DO NOT click on “Rebalance” just yet... Instead switch over to the App Server cmd line... the next few commands will explain why a rebalance is required at this point.

6 Node Cluster > Servers

Dashboard
Servers
Buckets
XDCR
Security
Settings
Logs
Documents
Query
Indexes
Search
Analytics
Eventing
Views

filter servers...

name ▼	group	services	CPU	RAM	swap	disk used	item
ec2-18-236-106-89.us-west-2.compute.amazonaws.com	Fall Domain 1(RACK 1)	data	3.0%	28.7%	---	1.03MB	0/0
ec2-34-220-16-142.us-west-2.compute.amazonaws.com	Fall Domain 1(RACK 1)	data	3.0%	29.3%	---	1.03MB	0/0
ec2-34-221-220-171.us-west-2.compute.amazonaws.com	Fall Domain 2(RACK 2)	data	3.5%	29.4%	---	1.03MB	0/0
ec2-35-165-112-69.us-west-2.compute.amazonaws.com	Fall Domain 2(RACK 2)	data	4.5%	30.8%	---	1.03MB	0/0
ec2-52-26-92-182.us-west-2.compute.amazonaws.com	Fall Domain 1(RACK 1)	query index search	1.0%	28.3%	---	---	0/0
ec2-52-27-57-168.us-west-2.compute.amazonaws.com	Fall Domain 2(RACK 2)	query index search	4.5%	30.0%	---	---	0/0

Warning: Rebalance required, some data is not currently replicated.

Run the next set of commands from the App Server (Black VM):

Once again, run the cbstats command against the 1st node’s public hostname and grep for active_num and replica_num:



Lab-4: Removing nodes, failover, misc, page 30

```
[ec2-user@AppServer ~]$ cbstats $NODE1:11210 all -u Administrator -p couchbase -b default | grep active_num
vb_active_num: 256
vb_active_num_non_resident: 0
[ec2-user@AppServer ~]$ cbstats $NODE1:11210 all -u Administrator -p couchbase -b default | grep replica_num
vb_replica_num: 0
vb_replica_num_non_resident: 0
```

Notice that the 1st node is still only hosting 256 active vBuckets and no replica vBuckets.

Prior to the rebalance is occurring, try running the cbstats command against the 1st node in a loop and you should see the # of replica vBuckets on this node increase slowly to a max of 256 vBuckets:

```
[ec2-user@AppServer ~]$ while true; do cbstats $NODE1:11210 all -u Administrator -p couchbase -b default | grep replica_num; sleep 2; done
```

Return to the Couchbase Web UI.

Click on **Rebalance** to create the replica vBuckets:

name	group	services	CPU	RAM	swap	disk used	items	
ec2-18-236-106-89.us-west-2.compute.amazonaws.com	Fail Domain 1(RACK 1)	data	3.0%	28.7%	---	1.03MB	0/0	Statistics
ec2-34-220-16-142.us-west-2.compute.amazonaws.com	Fail Domain 1(RACK 1)	data	3.0%	29.3%	---	1.03MB	0/0	Statistics
ec2-34-221-220-171.us-west-2.compute.amazonaws.com	Fail Domain 2(RACK 2)	data	3.5%	29.4%	---	1.03MB	0/0	Statistics
ec2-35-165-112-69.us-west-2.compute.amazonaws.com	Fail Domain 2(RACK 2)	data	4.5%	30.8%	---	1.03MB	0/0	Statistics
ec2-52-26-92-182.us-west-2.compute.amazonaws.com	Fail Domain 1(RACK 1)	query index search	1.0%	28.3%	---	---	0/0	Statistics
ec2-52-27-57-168.us-west-2.compute.amazonaws.com	Fail Domain 2(RACK 2)	query index search	4.5%	30.0%	---	---	0/0	Statistics

The rebalance operation should take about 1 – 2 minutes to complete:

name	group	services	CPU	RAM	swap	disk used	items	
ec2-18-236-106-89.us-west-2.compute.amazonaws.com	Fail Domain 1(RACK 1)	data	5.0%	28.8%	---	---	---	Statistics
ec2-34-220-16-142.us-west-2.compute.amazonaws.com	Fail Domain 1(RACK 1)	data	3.5%	29.7%	---	---	---	Statistics
ec2-34-221-220-171.us-west-2.compute.amazonaws.com	Fail Domain 2(RACK 2)	data	5.0%	29.6%	---	---	---	Statistics
ec2-35-165-112-69.us-west-2.compute.amazonaws.com	Fail Domain 2(RACK 2)	data	4.0%	30.8%	---	---	---	Statistics
ec2-52-26-92-182.us-west-2.compute.amazonaws.com	Fail Domain 1(RACK 1)	query index search	1.5%	28.6%	---	---	0/0	Statistics
ec2-52-27-57-168.us-west-2.compute.amazonaws.com	Fail Domain 2(RACK 2)	query index search	0.5%	30.0%	---	---	0/0	Statistics

Observe the 1st node in the loop and you should see the # of replica vBuckets on this node increase slowly to a max of 256 vBuckets:

Typed in earlier.....



Lab-4: Removing nodes, failover, misc, page 31

```
[ec2-user@AppServer ~]$ while true; do cbstats $NODE1:11210 all -u Administrator -p couchbase -b default | grep replica_num; sleep 2; done
```

```
vb_replica_num_non_resident: 0
vb_replica_num: 5
vb_replica_num_non_resident: 0
vb_replica_num: 42
vb_replica_num_non_resident: 0
vb_replica_num: 81
vb_replica_num_non_resident: 0
vb_replica_num: 122
vb_replica_num_non_resident: 0
vb_replica_num: 163
vb_replica_num_non_resident: 0
vb_replica_num: 201
vb_replica_num_non_resident: 0
vb_replica_num: 252
vb_replica_num_non_resident: 0
vb_replica_num: 256
vb_replica_num_non_resident: 0
vb_replica_num: 256
vb_replica_num_non_resident: 0
vb_replica_num: 256
```

You will finally see **256 replica vBuckets** on the 1st node.

Once the Rebalance operation is complete, the orange Warning message on the Servers page will disappear. Also notice that currently, 2 data nodes are under Server Group 1 and 2 data nodes are under Server Group 2(not counting the Query and index service nodes):

6 Node Cluster > Servers									
<div>FILTER GROUPS ADD SERVER</div> <div>Rebalance</div>									
name	group	services	CPU	RAM	swap	disk used	items		
ec2-13-56-207-54.us-west-1.compute.amazonaws.com	Group 2	data	1.01%	25.3%	---	6.06MB	0/0	Statistics	
ec2-13-57-48-149.us-west-1.compute.amazonaws.com	Group 2	data	2.01%	15.8%	---	6.05MB	0/0	Statistics	
ec2-52-53-173-24.us-west-1.compute.amazonaws.com	Group 1	data	2%	18.5%	---	6.07MB	0/0	Statistics	
ec2-54-183-85-83.us-west-1.compute.amazonaws.com	Group 1	data	2.51%	24%	---	6.07MB	0/0	Statistics	
ec2-54-193-62-173.us-west-1.compute.amazonaws.com	Group 2	full text index query	1.5%	17.5%	---	---	0/0	Statistics	
ec2-54-193-79-108.us-west-1.compute.amazonaws.com	Group 1	full text index query	1.49%	18%	---	---	0/0	Statistics	

Before we actually trigger a failover, let's write 10 keys to the default bucket.

But where will the 10 keys be distributed within the 4-node cluster's active vBuckets?



Lab-4: Removing nodes, failover, misc, page 32

The following `cbc hash` command's output will display 10 lines, each line will identify which of the 4-data-service nodes in the cluster the key will be directed to if a write were to occur(using the `cbc-create` command in the following lab steps).

Run this command on the App Server (black VM) against the 1st node's public IP(remember we don't have to pass a userID, password or IP address since it was set earlier):

```
[ec2-user@AppServer ~]$ cbc-hash key-1 key-2 key-3 key-4 key-5 key-6
key-7 key-8 key-9 key-10
```

```
key-1: [vBucket=748, Index=2] Server: ec2-54-172-130-69.compute-1.amazonaws.com:11210, CouchAPI:
http://ec2-54-172-130-69.compute-1.amazonaws.com:8092/default
Replica #0: Index=1, Host=ec2-54-172-130-66.compute-1.amazonaws.com:11210
```

```
key-2: [vBucket=997, Index=3] Server: ec2-54-172-130-8.compute-1.amazonaws.com:11210, CouchAPI:
http://ec2-54-172-130-8.compute-1.amazonaws.com:8092/default
Replica #0: Index=1, Host=ec2-54-172-130-66.compute-1.amazonaws.com:11210
```

```
key-3: [vBucket=226, Index=1] Server: ec2-54-172-130-66.compute-1.amazonaws.com:11210, CouchAPI:
http://ec2-54-172-130-66.compute-1.amazonaws.com:8092/default
Replica #0: Index=3, Host=ec2-54-172-130-8.compute-1.amazonaws.com:11210
```

```
key-4: [vBucket=646, Index=2] Server: ec2-54-172-130-69.compute-1.amazonaws.com:11210, CouchAPI:
http://ec2-54-172-130-69.compute-1.amazonaws.com:8092/default
Replica #0: Index=1, Host=ec2-54-172-130-66.compute-1.amazonaws.com:11210
```

```
key-5: [vBucket=385, Index=0] Server: ec2-54-172-130-15.compute-1.amazonaws.com:11210, CouchAPI:
http://ec2-54-172-130-15.compute-1.amazonaws.com:8092/default
Replica #0: Index=3, Host=ec2-54-172-130-8.compute-1.amazonaws.com:11210
```

```
key-6: [vBucket=136, Index=1] Server: ec2-54-172-130-66.compute-1.amazonaws.com:11210, CouchAPI:
http://ec2-54-172-130-66.compute-1.amazonaws.com:8092/default
Replica #0: Index=3, Host=ec2-54-172-130-8.compute-1.amazonaws.com:11210
```

```
key-7: [vBucket=911, Index=3] Server: ec2-54-172-130-8.compute-1.amazonaws.com:11210, CouchAPI:
http://ec2-54-172-130-8.compute-1.amazonaws.com:8092/default
Replica #0: Index=1, Host=ec2-54-172-130-66.compute-1.amazonaws.com:11210
```

```
key-8: [vBucket=816, Index=3] Server: ec2-54-172-130-8.compute-1.amazonaws.com:11210, CouchAPI:
http://ec2-54-172-130-8.compute-1.amazonaws.com:8092/default
Replica #0: Index=0, Host=ec2-54-172-130-15.compute-1.amazonaws.com:11210
```

```
key-9: [vBucket=55, Index=1] Server: ec2-54-172-130-66.compute-1.amazonaws.com:11210, CouchAPI:
http://ec2-54-172-130-66.compute-1.amazonaws.com:8092/default
Replica #0: Index=2, Host=ec2-54-172-130-69.compute-1.amazonaws.com:11210
```

```
key-10: [vBucket=5, Index=1] Server: ec2-54-172-130-66.compute-1.amazonaws.com:11210, CouchAPI:
http://ec2-54-172-130-66.compute-1.amazonaws.com:8092/default
Replica #0: Index=2, Host=ec2-54-172-130-69.compute-1.amazonaws.com:11210
```

Note that running the above command against any of the 4 nodes' public hostnames would result in the same, identical output.

Since we have Server Groups configured, we should see that if active data lands one of the first 2 nodes (Group 1), its replica ends up on one of the second 2 nodes (Group 2).

In other words, in my specific lab environment, I have the following two nodes configured in **Server Group #1**:



Node #1: ec2-54-85-43-128

Node #2: ec2-54-86-106-120

Server Group #2 contains:

Node #3: ec2-54-86-243-136

Node #4: ec2-54-85-206-193

So, in my example, any of the active data that lands in Node #1 or #2, should have its replica stored on Node #3 or #4.

If you scroll back up 1 or 2 pages to the output of where the 10 keys will be stored, this pattern should now become more apparent.

For example, in my environment:

key-1: Active: Node #4 / Replica: Node #2

key-2: Active: Node # 4 / Replica: Node #2

key-3: Active: Node # 1 / Replica: Node #3

key-4: Active: Node # 2 / Replica: Node #3

key-5: Active: Node # 2 / Replica: Node #4

key-6: Active: Node # 1 / Replica: Node #3

key-7: Active: Node # 3 / Replica: Node #2

key-8: Active: Node # 3 / Replica: Node #2

key-9: Active: Node # 1 / Replica: Node #4

key-10: Active: Node # 1 / Replica: Node #4

Notice above that Node #1 is holding 4 Active keys and 0 Replica keys.

At this time, please take a few minutes to make a quick document similar to the 10 lines I have above using your cluster's custom configuration. You should also see in your specific lab environment, that replicas for the active data are being placed in a different Server Group.



Lab-4: Removing nodes, failover, misc, page 34

Switch to the App Server (black VM) and insert the following 10 keys with 10 values into the cluster. You can use the public hostname of any of the 4 nodes in the next two pages for the 'cbc create' and 'cbc cat' commands (*note, you may need to hit CTRL + D a few times below before it takes effect*):

```
[ec2-user@AppServer ~]$ for num in {1..10}; do cbc-create key-$num -V
{"value\":\":${num}\"}; done
```

key-1	Stored. CAS=0x15a3749fab370000
key-2	Stored. CAS=0x15a3749fac0f0000
key-3	Stored. CAS=0x15a3749fac9a0000
key-4	Stored. CAS=0x15a3749fad6a0000
key-5	Stored. CAS=0x15a3749fae410000
key-6	Stored. CAS=0x15a3749faef30000
key-7	Stored. CAS=0x15a3749fafb10000
key-8	Stored. CAS=0x15a3749fb0640000
key-9	Stored. CAS=0x15a3749fb1210000
key-10	Stored. CAS=0x15a3749fb1e00000

Switch to the Web UI and **click on Data Buckets** at the top to verify that there are now 10 items in the default bucket.

6 Node Cluster > Buckets							ADD BUCKET
name	items	resident	ops/sec	RAM used/quota	disk used		
default	10	100%	0	46.9MB / 7.1GB	24.3MB	Documents	Statistics

Documents !

Then **click on Documents**:

On the next page, you should see all 10 keys sorted in lexicographical order(change page view to 10):



Lab-4: Removing nodes, failover, misc, page 35

6 Node Cluster > Documents			
Dashboard	Bucket	show top keys	Limit 10
Servers	default	Offset 0	Document ID optional...
Buckets			show range NTQL WHERE no indexes available...
XDCR			
Security			
Settings			
Logs			
Documents			
Query			
Indexes			
Search			
Analytics			
Eventing			
Views			

10 Results for default, limit: 10, offset: 0			enable field editing
key	value	key	value
key-1	["value":"1"]	key-10	["value":"10"]
key-2	["value":"2"]	key-3	["value":"3"]
key-4	["value":"4"]	key-5	["value":"5"]
key-6	["value":"6"]	key-7	["value":"7"]
key-8	["value":"8"]	key-9	["value":"9"]

Switch back to the App Server (black VM) and check the Auto-failover settings on the default bucket via the REST API (you can use the public hostname of any of the 4 nodes in this command):

```
[ec2-user@AppServer ~]$ curl -u Administrator:couchbase -n
http://$NODE1:8091/settings/autoFailover
{"enabled":true,"timeout":120,"count":0,"failoverOnDataDiskIssues":{"e
nabled":false,"timePeriod":120},"failoverServerGroup":false,"maxCount"
:1}
```

The following parameters and settings appear in the output above:

enabled : either true if auto-failover is enabled or false if it is not.

timeout : seconds that must elapse before auto-failover executes on a cluster.

count : can be 0,1,2, or 3. Number of times any node in a cluster can be automatically failed-over. After one auto-failover occurs, count is set to 1

maxCount : can be set to 1,2 or 3

The following command disables auto-failover and then next verifies that it is disabled:

Note : The default setting of enabled

```
[ec2-user@appserver ~]$ curl -u Administrator:couchbase -n
http://$NODE1:8091/settings/autoFailover -i -d 'enabled=false'
HTTP/1.1 200 OK
X-XSS-Protection: 1; mode=block
X-Permitted-Cross-Domain-Policies: none
X-Frame-Options: DENY
X-Content-Type-Options: nosniff
Server: Couchbase Server
Pragma: no-cache
Expires: Thu, 01 Jan 1970 00:00:00 GMT
Date: Tue, 19 Mar 2019 17:12:47 GMT
Content-Length: 0
Cache-Control: no-cache,no-store,must-revalidate
```



Lab-4: Removing nodes, failover, misc, page 36

```
[ec2-user@appserver ~]$ curl -u Administrator:couchbase -n
http://$NODE1:8091/settings/autoFailover
{"enabled":false,"timeout":120,"count":0,"failoverOnDataDiskIssues":{"enabled":false,"timePeriod":120},"failoverServerGroup":false,"maxCount":1}
```

Note that the autoFailover settings can also be configured using the Couchbase CLI as described here:

<http://docs.couchbase.com/admin/admin/CLI/CBcli/cbcli-commands.html>

Switch back to the Web UI and **click on Servers** :

name	group	services	CPU	RAM	swap	disk used	items
ec2-13-56-207-54.us-west-1.compute.amazonaws.com	Group 2	data	1.51%	25.4%	---	6.08MB	4/1
ec2-13-57-48-149.us-west-1.compute.amazonaws.com	Group 2	data	2.5%	15.8%	---	6.07MB	2/3
ec2-52-53-173-24.us-west-1.compute.amazonaws.com	Group 1	data	1%	18.4%	---	6.09MB	2/2
ec2-54-183-85-83.us-west-1.compute.amazonaws.com	Group 1	data	5%	24.1%	---	6.1MB	2/4
ec2-54-193-62-173.us-west-1.compute.amazonaws.com	Group 2	full text index query	1.5%	17.1%	---	---	0/0
ec2-54-193-79-108.us-west-1.compute.amazonaws.com	Group 1	full text index query	2.01%	17.9%	---	---	0/0

In the column for Items (Active/Replica) you should see the 10 keys that we created and 10 replica keys for them also, so the total count should be 20.

The numbers that you see here should correlate to the chart that you created for the 10 keys earlier in this lab.

For example, Node #1 in my cluster starts with the Public hostname “ec2-54-85-43-128”, which also just happens to be the 1st node in the screenshot above. This node is holding 4 active keys and 0 replica keys. This matches what I was expecting after creating the chart earlier. Perform the same check for a few of the nodes in your environment.

Next we will stop the Couchbase service on both nodes in Server Group 2. Note by looking at the screenshot below, in my specific cluster, I will lose 6 active keys by doing this (4 active keys on one node and 2 active on the other node).

name	group	services	CPU	RAM	swap	disk used	items
ec2-13-56-207-54.us-west-1.compute.amazonaws.com	Group 2	data	1.5%	25.3%	---	6.08MB	4/1
ec2-13-57-48-149.us-west-1.compute.amazonaws.com	Group 2	data	3%	15.8%	---	6.07MB	2/3



More specifically for my specific installation, active key-4 and key-6 will no longer be available on Node #3 and active key-1 and key-2 will no longer be available on Node #4.

After stopping the Couchbase process on Nodes 3 and 4, we will need to failover Node 3 and Node 4 before the replicas get activated and we can read those 4 keys from the activated replicas.

Warning: The keys that you will need to test for your specific installation will differ from the keys that I will be using. Use the chart that you created earlier to figure out which keys are active on Node 3 and Node 4 and those are the keys that you will be reading below. You may want to write down which active keys are hosted on Node 3 and which active keys are hosted on Node 4 before continuing:

Fill out this form:

Active Keys hosted on Node #3: _____

Active Keys hosted on Node # 4: _____

First read an active key from Node #3. Use the form you filled right above to pick a key from Node #3. In my specific lab, I have picked key-6. **You may have to pick a different key.** Run the following command from the App Server (use the 1st node's public hostname for the command below):

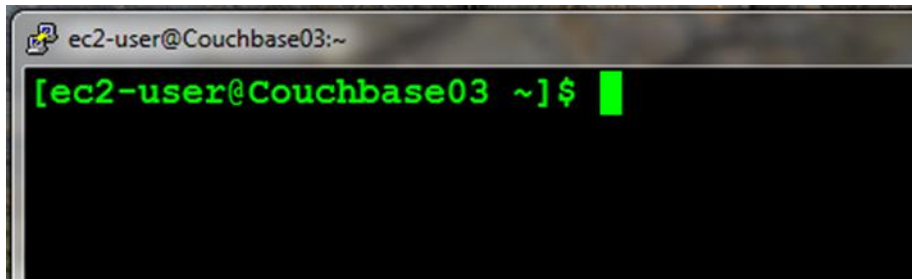
```
[ec2-user@AppServer ~]$ cbc-cat -u Administrator -P couchbase -U  
couchbase://ec2-54-172-130-66.compute-1.amazonaws.com/default key-6  
key-6 CAS=0x1da067b2e1920400, Flags=0x0, Datatype=0x0  
value-6
```

So far so good.

Next, switch to the 3rd node's (Green VM/Couchbase03) PuTTY window and stop the Couchbase service on the node:



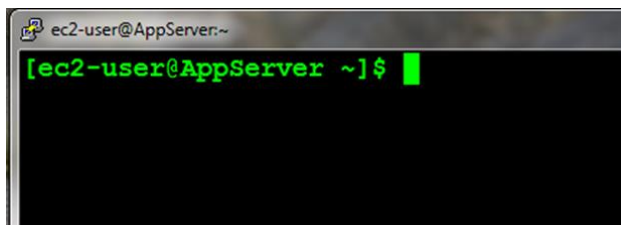
Lab-4: Removing nodes, failover, misc, page 38



```
[ec2-user@Couchbase03 ~]$ sudo systemctl stop couchbase-server
Stopping couchbase-server
```

```
[ec2-user@Couchbase03 ~]$ sudo systemctl status couchbase-server
Aug 10 13:37:11 node3 systemd[1]: Stopped LSB: couchbase server.
```

Switch over to the App Server (black VM) and try to read the key you just read from Node #3 again. I will attempt reading key-7 again for my specific environment. All of the following commands will be run from the App Server, until noted otherwise:



```
[ec2-user@AppServer ~]$ cbc-cat -U $NODE3/default key-6
Failed to bootstrap instance. libcouchbase error: LCB_ECONNREFUSED (0x2C):
The remote host refused the connection. Is the service up?
```

or you can re-run the previous cbc-cat command from your buffer(command history)

NOTE: this is using \$NODE1 variable setting from your .cbcrc file

```
cbc-cat key-1 key-2 key-3 key-4 key-5 key-6 key-7 key-8 key-9 key-10
```

```
key-5          CAS=0x158dbe3cb8ce0000, Flags=0x0, Size=7, Datatype=0x00
value-5
key-9          CAS=0x158dbe4c2b3f0000, Flags=0x0, Size=7, Datatype=0x00
value-9
key-10         CAS=0x158dbe4f7deb0000, Flags=0x0, Size=8, Datatype=0x00
value-10
key-2          CAS=0x158dbe2f32620000, Flags=0x0, Size=7, Datatype=0x00
value-2
key-7          CAS=0x158dbe43625d0000, Flags=0x0, Size=7, Datatype=0x00
value-7
key-8          CAS=0x158dbe46c9f60000, Flags=0x0, Size=7, Datatype=0x00
value-8
key-1          CAS=0x158dbe1058080000, Flags=0x0, Size=7, Datatype=0x00
value-1
```



Lab-4: Removing nodes, failover, misc, page 39

```
key-3          CAS=0x158dbe323e320000, Flags=0x0, Size=7, Datatype=0x00
value-3
key-4          LCB_ECONNREFUSED (0x2C)
key-6          LCB_ECONNREFUSED (0x2C)
```

Reading key-4 will also fail on my cluster since that was another key located on Node #3:

```
[ec2-user@AppServer ~]$ cbc-cat -U $NODE3/default key-4
```

Failed to bootstrap instance. libcouchbase error: LCB_ECONNREFUSED (0x2C):
The remote host refused the connection. Is the service up?

However reading key-1 and key-2 will work successfully, since those keys are stored on Node #4 in my environment. Run a similar test in your environment to ensure that you can still read the active keys from Node #4:

```
[ec2-user@AppServer ~]$ cbc-cat -U $NODE4/default key-1 key-2 key-3
key-4 key-5 key-6 key-7 key-8 key-9 key-10
key-2          CAS=0x158dbe2f32620000, Flags=0x0, Size=7, Datatype=0x00
value-2
key-7          CAS=0x158dbe43625d0000, Flags=0x0, Size=7, Datatype=0x00
value-7
key-8          CAS=0x158dbe46c9f60000, Flags=0x0, Size=7, Datatype=0x00
value-8
key-1          CAS=0x158dbe1058080000, Flags=0x0, Size=7, Datatype=0x00
value-1
key-3          CAS=0x158dbe323e320000, Flags=0x0, Size=7, Datatype=0x00
value-3
key-5          CAS=0x158dbe3cb8ce0000, Flags=0x0, Size=7, Datatype=0x00
value-5
key-9          CAS=0x158dbe4c2b3f0000, Flags=0x0, Size=7, Datatype=0x00
value-9
key-10         CAS=0x158dbe4f7deb0000, Flags=0x0, Size=8, Datatype=0x00
value-10
key-4          LCB_ECONNREFUSED (0x2C)
key-6          LCB_ECONNREFUSED (0x2C)
```

Switch over to the Web UI and under Server Nodes, you should see the 3rd node marked as Down. Click on Fail Over for the 3rd/Down node:

name	group	services	CPU	RAM	swap	disk used	items	
ec2-13-56-207-54.us-west-1.compute.amazonaws.com	Group 2	data	1.51%	24.3%	---	11.2MB	4/1	Statistics
ec2-13-57-48-149.us-west-1.compute.amazonaws.com	Group 2	data	14.1%	16.1%	---	9.13MB	2/3	
Node unresponsive Not taking traffic FAILOVER to activate available replicas								Failover
ec2-52-53-173-24.us-west-1.compute.amazonaws.com	Group 1	data	1.99%	18.6%	---	10.7MB	2/2	Statistics
ec2-54-183-85-83.us-west-1.compute.amazonaws.com	Group 1	data	2.5%	24.2%	---	11.9MB	2/4	Statistics
ec2-54-193-62-173.us-west-1.compute.amazonaws.com	Group 2	full text index query	2.51%	17.9%	---	---	0/0	Statistics
ec2-54-193-79-108.us-west-1.compute.amazonaws.com	Group 1	full text index query	1.52%	18.3%	---	---	0/0	Statistics

On the Confirm Node Failover popup, place a check mark next to “Please confirm Failover” and click on Fail Over:



Confirm Failover - ec2-35-165-112-69.us-west-2.compute.amazonaws.com:8091
X

Failover Options

☒ Graceful Failover (default)

☐ Hard Failover

Cancel
Failover Node

Run cbstats against all four data service nodes(Couch01,02,04 and 03 last)

```
[ec2-user@appserver ~]$ cbstats $NODE1:11210 all -u
Administrator -p couchbase -b default | grep active_num
vb_active_num: 384
vb_active_num_non_resident: 0
```

```
[ec2-user@appserver ~]$ cbstats $NODE2:11210 all -u
Administrator -p couchbase -b default | grep active_num
vb_active_num: 384
vb_active_num_non_resident: 0
```

```
[ec2-user@appserver ~]$ cbstats $NODE4:11210 all -u
Administrator -p couchbase -b default | grep active_num
vb_active_num: 256
vb_active_num_non_resident: 0
```

```
[ec2-user@appserver ~]$ cbstats $NODE1:11210 all -u
Administrator -p couchbase -b default | grep replica_num
vb_replica_num: 128
vb_replica_num_non_resident: 0
```

```
[ec2-user@appserver ~]$ cbstats $NODE2:11210 all -u
Administrator -p couchbase -b default | grep replica_num
vb_replica_num: 128
vb_replica_num_non_resident: 0
```

```
[ec2-user@appserver ~]$ cbstats $NODE4:11210 all -u
Administrator -p couchbase -b default | grep replica_num
vb_replica_num: 256
vb_replica_num_non_resident: 0
```

```
[ec2-user@appserver ~]$ cbstats $NODE3:11210 all -u
Administrator -p couchbase -b default | grep active_num
Exception AttributeError: "'MemcachedClient' object has no
attribute 's'" in <bound method MemcachedClient.__del__ of
```




Lab-4: Removing nodes, failover, misc, page 41

```
<mc_bin_client.MemcachedClient object at 0x7f4ceb83f910>>
ignored
No response.
```

Note: Couch03 couchservice is dead

Do the math..... how many active vbuckets are there? (1024)

How many replica vbuckets are there? (512)

Why?

Run the following command.

```
[ec2-user@AppServer ~]$ cbc-cat key-1 key-2 key-3 key-4 key-5
key-6 key-7 key-8 key-9 key-10
key-1          CAS=0x158dbe1058080000, Flags=0x0, Size=7, Datatype=0x00
value-1
key-3          CAS=0x158dbe323e320000, Flags=0x0, Size=7, Datatype=0x00
value-3
key-6          CAS=0x158dbe4065800000, Flags=0x0, Size=7, Datatype=0x00
value-6
key-4          CAS=0x158dbe37fe890000, Flags=0x0, Size=7, Datatype=0x00
value-4
key-5          CAS=0x158dbe3cb8ce0000, Flags=0x0, Size=7, Datatype=0x00
value-5
key-9          CAS=0x158dbe4c2b3f0000, Flags=0x0, Size=7, Datatype=0x00
value-9
key-10         CAS=0x158dbe4f7deb0000, Flags=0x0, Size=8, Datatype=0x00
value-10
key-2          CAS=0x158dbe2f32620000, Flags=0x0, Size=7, Datatype=0x00
value-2
key-7          CAS=0x158dbe43625d0000, Flags=0x0, Size=7, Datatype=0x00
value-7
key-8          CAS=0x158dbe46c9f60000, Flags=0x0, Size=7, Datatype=0x00
value-8
```

notice it now returns all 10 keys?

Summary. You can read all ten keys now but there is no backup if there was a second failure of another node.

A Rebalance will now be required. Click on "Rebalance":



Lab-4: Removing nodes, failover, misc, page 42

name	group	services	CPU	RAM	swap	disk used	items	
ec2-13-56-207-54.us-west-1.compute.amazonaws.com	Group 2	data	1%	23.9%	---	11.2MB	4/1	Statistics
ec2-13-57-48-149.us-west-1.compute.amazonaws.com	Group 2	data	14.1%	16.1%	---	9.13MB	2/3	
Node unresponsive, failed-over Not taking traffic REMOVAL pending rebalance								
ec2-52-53-173-24.us-west-1.compute.amazonaws.com	Group 1	data	1.5%	18.5%	---	9.98MB	2/2	Statistics
ec2-54-183-85-83.us-west-1.compute.amazonaws.com	Group 1	data	2.5%	23.9%	---	10MB	4/2	Statistics
ec2-54-193-62-173.us-west-1.compute.amazonaws.com	Group 2	full text index query	1.01%	17.2%	---	---	0/0	Statistics
ec2-54-193-79-108.us-west-1.compute.amazonaws.com	Group 1	full text index query	0.5%	17.6%	---	---	0/0	Statistics

The Rebalance will now start and will take about 2 – 3 minutes to complete and you should see only 5 nodes in the cluster:

6 Node Cluster > Servers

rebalancing 5 nodes 9.1%

name	group	services	CPU	RAM	swap	disk used	items
ec2-13-56-207-54.us-west-1.compute.amazonaws.com	Group 2	data	4.04%	24.1%	---	11.2MB	4/1
ec2-52-53-173-24.us-west-1.compute.amazonaws.com	Group 1	data	1%	18.5%	---	9.98MB	2/2
ec2-54-183-85-83.us-west-1.compute.amazonaws.com	Group 1	data	2.51%	23.9%	---	10MB	4/2
ec2-54-193-62-173.us-west-1.compute.amazonaws.com	Group 2	full text index query	1%	16.9%	---	---	0/0
ec2-54-193-79-108.us-west-1.compute.amazonaws.com	Group 1	full text index query	1%	17.5%	---	---	0/0

After the Rebalance query for the active keys that were on Node #3 (key-4 and key-6 in my environment) and you should successfully see them again since their replicas have been activated:

```
[ec2-user@AppServer~]$ cbc-cat -u Administrator -P couchbase -U couchbase://ec2-54-172-130-66.compute-1.amazonaws.com/default key-4
key-7
value-7
CAS=0x1da067b2e1920400, Flags=0x0, Size=7
```

Reading key-8 will also work on my cluster since this was another key located on Node #3 and now has its replica activated:

```
[ec2-user@AppServer ~]$ cbc-cat -u Administrator -P couchbase -U couchbase://ec2-54-172-130-66.compute-1.amazonaws.com/default key-6
key-8
value-8
CAS=0x713bce5b3e930400, Flags=0x0, Size=7
```

By looking at the chart I created earlier, I can assume that the key-4 and key-6 are being served from Node #2 as this was the node holding the replica for the data. Here is the chart reprinted for my environment:

key-1: Active: Node #4 / Replica: Node #2

key-2: Active: Node #4 / Replica: Node #2

key-3: Active: Node #1 / Replica: Node #3



key-4: Active: Node # 2 / Replica: Node #3

key-5: Active: Node # 2 / Replica: Node #4

key-6: Active: Node # 1 / Replica: Node #3

key-7: Active: Node # 3 / Replica: Node #2

key-8: Active: Node # 3 / Replica: Node #2

key-9: Active: Node # 1 / Replica: Node #4

key-10: Active: Node # 1 / Replica: Node #4

Look at the chart for your environment and figure out which node would be hosting the replicas for the active keys that were originally on Node #3.

Check which nodes are hosting those keys now (I will check for key-4 and key-6 myself):

```
[ec2-user@AppServer ~]$ cbc-hash key-4 key-6
```

```
key-4: [vBucket=646, Index=2] Server: ec2-18-144-49-199.us-west-1.compute.amazonaws.com:11210, CouchAPI: http://ec2-18-144-49-199.us-west-1.compute.amazonaws.com:8092/default
Replica #0: Index=0, Host=ec2-13-56-188-91.us-west-1.compute.amazonaws.com:11210
```

```
key-6: [vBucket=136, Index=0] Server: ec2-13-56-188-91.us-west-1.compute.amazonaws.com:11210, CouchAPI: http://ec2-13-56-188-91.us-west-1.compute.amazonaws.com:8092/default
Replica #0: Index=1, Host=ec2-13-56-209-124.us-west-1.compute.amazonaws.com:11210
```

For my cluster, the above output translates to:

key-4: Active: Node # 2 / Replica: Node #1

key-6: Active: Node # 2 / Replica: Node #1

At this point, it makes sense that Node #2 is hosting the active replica.

You may find it odd that the Replicas for key-4 and key-6 are may be hosted in the same server group as the active data (both are in Group #1 in my specific environment, but this may not be true for your environment). This can occur in very small Couchbase clusters which have an uneven number of nodes in each server group. The best practice in Couchbase is to replace a failed hardware node in a server group with a working node and *then* run rebalance across all the nodes.

Notice that in the 'cbc-hash' command above, key-4's active copy is currently in vBucket 646 on the server with the public hostname starting with 'ec2-18-144-49-199.us-west', which translates to Node #4 in my environment (it may be on another machine in your specific lab).



Finding key-4's active replica in your cluster:

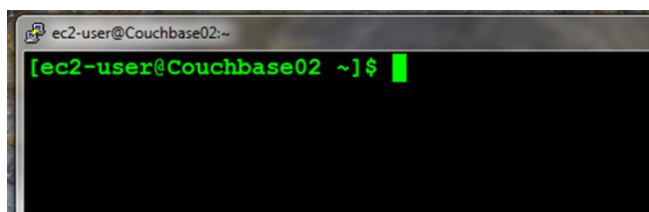
Let's try to find the data file containing key-4 in the Couchbase cluster. Regardless of which keys you were working with (that were originally active on Node #3) in the previous section, in this section try to find key-4.

Run the following cbc hash command from the App Server (black VM) against any node's public hostname in the cluster to find where key-4 is active:

```
[ec2-user@AppServer ~]$ cbc-hash key-4
```

```
key-4: [vBucket=646, Index=2] Server: ec2-18-144-49-199.us-west-1.compute.amazonaws.com:11210, CouchAPI: http://ec2-18-144-49-199.us-west-1.compute.amazonaws.com:8092/default
Replica #0: Index=0, Host=ec2-13-56-188-91.us-west-1.compute.amazonaws.com:11210
```

Since key-4 appears to be on **Node #4** in my environment in **vBucket 646**, I will switch to the PuTTY window for Node #4 (Couchbase04/Light Blue PuTTY window) to verify that key-4 is indeed on Node #4.



Warning: For your specific lab, switch over to the node where the Active copy of key-4 is located. This may not be Node #4 in your environment!

Let's go to the node where key-4 is located and actually locate it in a vBucket data file.

First switch to the root user:

```
[ec2-user@Couchbase02 ~]$ sudo -s
```

Change directories to the default bucket's data files:

```
[root@Couchbase02 ec2-user]# cd /opt/couchbase/var/lib/couchbase/data/default/
```

Search for **vBucket # 646** (note, in your environment, this may be a different numbered bucket!):

```
[root@Couchbase02 default]# ls | grep 646
646.couch.1
```



Once you've located the relevant bucket, print its contents with the `couch_dbdump` command:

```
[root@ Couchbase02 default]# /opt/couchbase/bin/couch_dbdump
./646.couch.1
```

```
Dumping "./646.couch.1":
Doc seq: 1
  id: key-4
  rev: 1
  content_meta: 131
  size (on disk): 17
  cas: 1553106594215559168, expiry: 0, flags: 0, datatype: 0x00 (raw)
  size: 7
  data: (snappy) value-4
```

```
Total docs: 1
```

Exit root:

```
[root@Couchbase02 default]# exit
exit
```

Excellent! You've successfully located key-4 in the underlying data files.

Next, run the `cbstats` command against the 1st 2nd, and 4th Node's public hostname to see how many vBuckets are currently assigned to it (this command should be run from the App Server/black node):

```
[ec2-user@appserver ~]$ cbstats $NODE1:11210 all -u
Administrator -p couchbase -b default | grep active_num
vb_active_num: 341
vb_active_num_non_resident: 0
[ec2-user@appserver ~]$ cbstats $NODE2:11210 all -u
Administrator -p couchbase -b default | grep active_num
vb_active_num: 341
vb_active_num_non_resident: 0
[ec2-user@appserver ~]$ cbstats $NODE4:11210 all -u
Administrator -p couchbase -b default | grep active_num
vb_active_num: 342
vb_active_num_non_resident: 0
```

After Checking on active vBucket numbers, check on replica numbers.



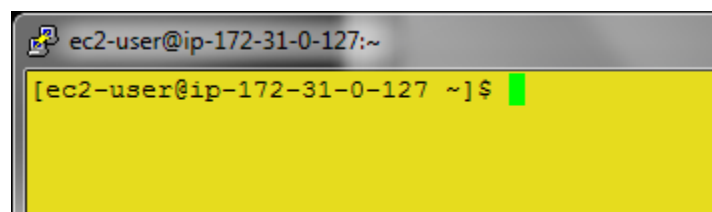
```
[ec2-user@appserver ~]$ cbstats $NODE1:11210 all -u
Administrator -p couchbase -b default | grep replica_num
vb_replica_num: 342
vb_replica_num_non_resident: 0
[ec2-user@appserver ~]$ cbstats $NODE2:11210 all -u
Administrator -p couchbase -b default | grep replica_num
vb_replica_num: 341
vb_replica_num_non_resident: 0
[ec2-user@appserver ~]$ cbstats $NODE4:11210 all -u
Administrator -p couchbase -b default | grep replica_num
vb_replica_num: 341
vb_replica_num_non_resident: 0
```

The 1st node in the cluster now has about 342 active vBuckets and 341 replica vBuckets buckets. This makes sense because $342 \text{ vBuckets} * 3 \text{ nodes} = 1,026$. There are a total of 1024 vBuckets in Couchbase, but when there is an odd number of nodes, Couchbase will make a best effort to distribute the vBuckets evenly. In this case, the other 2 remaining nodes will have 341 active vBuckets, so $342 + 341 + 341 = 1024$. The number of vBuckets you see on the 1st node in your environment could vary slightly.

Failover the 4th node in your cluster:

Finally, let's stop the Couchbase service on the 4th Node as well, so that all of the servers in Server Group #2 will be decommissioned.

Switch to the 4th node's (Yellow VM) PuTTY window and stop the Couchbase service on the node:



Run the following commands to stop Couchbase and check its status:

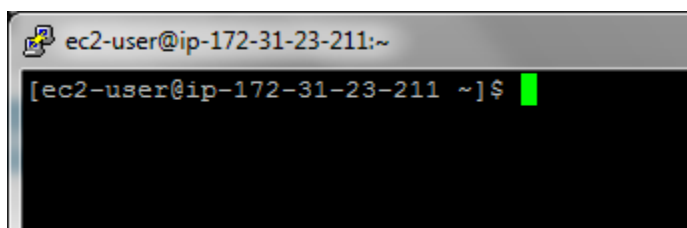
```
[ec2-user@Couchbase04 ~]$ sudo systemctl stop couchbase-server

[ec2-user@Couchbase04 ~]$ sudo systemctl status couchbase-server
Aug 10 13:53:09 node4 systemd[1]: Stopped LSB: couchbase server.
```



Lab-4: Removing nodes, failover, misc, page 47

This time, we'll fail over Node #4 using the Couchbase CLI. Switch to the App Server (black VM):



Run the following Couchbase CLI failover command to failover Node #4 (yellow VM). Note that in this command there are two public hostname's provided. For the --cluster hostname use the public hostname of **Node #1**. For the --server-failover hostname, provide the public hostname for **Node #4**:

```
[ec2-user@AppServer ~]$ couchbase-cli failover --cluster=$NODE1:8091 --server-failover=$NODE4:8091 --force
SUCCESS: Server failed over
```

FYI NOTE: This is what the command would look like if all variables had to be specified at the command line

```
[ec2-user@appserver ~]$ couchbase-cli failover --cluster=http://ec2-54-183-85-83.us-west-1.compute.amazonaws.com:8091 --server-failover=ec2-13-56-207-54.us-west-1.compute.amazonaws.com:8091 --user=Administrator --password=couchbase --force
```

Once you see a success message, switch over to the Web UI and you should see a Down node #4 with a Rebalance pending:

Warning: Rebalance required, some data is not currently replicated.								
								Rebalance
name	group	services	CPU	RAM	swap	disk used	items	
ec2-13-56-207-54.us-west-1.compute.amazonaws.com	Group 2	data	0%	---	---	13.9MB	2/3	
Node unresponsive, failed-over Not taking traffic REMOVAL pending rebalance								
ec2-52-53-173-24.us-west-1.compute.amazonaws.com	Group 1	data	2%	14.7%	---	10.4MB	3/5	Statistics
ec2-54-183-85-83.us-west-1.compute.amazonaws.com	Group 1	data	2.01%	23.8%	---	14MB	7/0	Statistics
ec2-54-193-62-173.us-west-1.compute.amazonaws.com	Group 2	full-text index query	2.5%	17%	---	---	0/0	Statistics
ec2-54-193-79-108.us-west-1.compute.amazonaws.com	Group 1	full-text index query	1.5%	17.7%	---	---	0/0	Statistics



Lab-4: Removing nodes, failover, misc, page 48

Switch back to the App Server's command line and trigger a Rebalance operation using the Couchbase CLI as well.

(Note that the `--cluster` parameter should provide the public hostname of **Node #1** and the `--server-remove` parameter should provide the public hostname for **Node #4** (the node we want to remove))

ALSO... Run next two commands with in a few seconds of each other.

One command on appserver and one command on Couchbase01

(you may want to type commands on each VM before executing them):

```
[ec2-user@appserver ~]$ couchbase-cli rebalance --cluster=$NODE1:8091
--server-remove=$NODE4:8091
```

Rebalancing

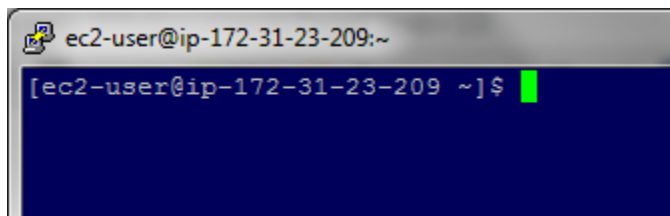
Bucket: 00/00 ()

0 docs remaining

```
[=====] 100.0%
```

SUCCESS: Rebalance complete

While the rebalance is running, quickly switch to Node #1 (dark blue) and run the following command to check the status of the Rebalance operation:



```
[ec2-user@couchbase01 ~]$ couchbase-cli rebalance-status
--cluster=$NODE1:8091
{
  "status": "running",
  "msg": "Rebalance is running",
  "details": {
    "totalBuckets": 1,
    "curBucket": 1,
    "refresh": 0.25,
    "docsRemaining": 14,
    "curBucketName": "default",
    "progress": 1.326701539572522
  }
}
```




```
[ec2-user@couchbase01 ~]$ couchbase-cli rebalance-status  
--cluster=$NODE1:8091
```

```
{  
  "status": "running",  
  "msg": "Rebalance is running",  
  "details": {  
    "totalBuckets": 1,  
    "curBucket": 1,  
    "refresh": 0.25,  
    "docsRemaining": 10,  
    "curBucketName": "default",  
    "progress": 10.73084561083574  
  }  
}
```

```
[ec2-user@couchbase01 ~]$ couchbase-cli rebalance-status  
--cluster=$NODE1:8091
```

```
{  
  "status": "running",  
  "msg": "Rebalance is running",  
  "details": {  
    "totalBuckets": 1,  
    "curBucket": 1,  
    "refresh": 0.25,  
    "docsRemaining": 6,  
    "curBucketName": "default",  
    "progress": 24.36932829664365  
  }  
}
```

```
[ec2-user@couchbase01 ~]$ couchbase-cli rebalance-status  
--cluster=$NODE1:8091
```

```
{  
  "status": "running",  
  "msg": "Rebalance is running",  
  "details": {  
    "totalBuckets": 1,  
    "curBucket": 1,  
    "refresh": 0.25,  
    "docsRemaining": 6,  
    "curBucketName": "default",  
    "progress": 33.09052558450966  
  }  
}
```

```
[ec2-user@couchbase01 ~]$ couchbase-cli rebalance-status  
--cluster=$NODE1:8091
```

```
{  
  "status": "running",  
  "msg": "Rebalance is running",  
  "details": {  
    "totalBuckets": 1,  
    "curBucket": 1,  
    "refresh": 0.25,  
    "docsRemaining": 6,  
    "curBucketName": "default",  
    "progress": 33.09052558450966  
  }  
}
```



Lab-4: Removing nodes, failover, misc, page 50

```

    "refresh": 0.25,
    "docsRemaining": 2,
    "curBucketName": "default",
    "progress": 42.37983587338805
  }
}
[ec2-user@couchbase01 ~]$ couchbase-cli rebalance-status
--cluster=$NODE1:8091
{
  "status": "notRunning",
  "msg": "Rebalance is not running",
  "details": {}
}

```

Great, the rebalance operation is definitely complete. The web UI should also reflect the change:

name	group	services	CPU	RAM	swap	disk used	items
ec2-52-53-173-24.us-west-1.compute.amazonaws.com	Group 1	data	1.5%	15.5%	---	19MB	5/5
ec2-54-183-85-83.us-west-1.compute.amazonaws.com	Group 1	data	2.98%	24.1%	---	22.7MB	5/5
ec2-54-193-62-173.us-west-1.compute.amazonaws.com	Group 2	full text index query	2.02%	17%	---	---	0/0
ec2-54-193-79-108.us-west-1.compute.amazonaws.com	Group 1	full text index query	1.01%	17.8%	---	---	0/0

Notice that in the Web UI, now there are 5 active items and 5 replica(possibly 6/4) items on each of the 2 remaining nodes in Server Group #1.

Run the `cbstats` command from the App Server (black VM) and provide the first Node's public hostname:

```

[ec2-user@AppServer ~]$ cbstats $NODE1:11210 all -u Administrator -p
couchbase -b default | grep active_num

```

```

vb_active_num:                    512
vb_active_num_non_resident:       0

```

```

[ec2-user@AppServer ~]$ cbstats $NODE1:11210 all -u Administrator -p
couchbase -b default | grep replica_num

```

```

vb_replica_num:                   512
vb_replica_num_non_resident:      0

```

You can see that the rebalance operation has placed 512 active vBuckets and 512 replica vBuckets on each of the remaining 2 nodes.



Finally, try querying for the 10 keys we originally inserted to verify that they are still available after a loss of an entire Server Group. You can run all of the remaining commands on the App Server and provide it with the 1st node's public hostname:

```
[ec2-user@appserver ~]$ cbc-cat key-1 key-2 key-3 key-4 key-5 key-6
key-7 key-8 key-9 key-10
```

```
key-3          CAS=0x158dbe323e320000, Flags=0x0, Size=7,
Datatype=0x00
value-3
key-4          CAS=0x158dbe37fe890000, Flags=0x0, Size=7,
Datatype=0x00
value-4
key-6          CAS=0x158dbe4065800000, Flags=0x0, Size=7,
Datatype=0x00
value-6
key-9          CAS=0x158dbe4c2b3f0000, Flags=0x0, Size=7,
Datatype=0x00
value-9
key-10         CAS=0x158dbe4f7deb0000, Flags=0x0, Size=8,
Datatype=0x00
value-10
key-1          CAS=0x158dbe1058080000, Flags=0x0, Size=7,
Datatype=0x00
value-1
key-2          CAS=0x158dbe2f32620000, Flags=0x0, Size=7,
Datatype=0x00
value-2
key-5          CAS=0x158dbe3cb8ce0000, Flags=0x0, Size=7,
Datatype=0x00
value-5
key-7          CAS=0x158dbe43625d0000, Flags=0x0, Size=7,
Datatype=0x00
value-7
key-8          CAS=0x158dbe46c9f60000, Flags=0x0, Size=7,
Datatype=0x00
value-8
```

```
[ec2-user@appserver ~]$ cbc-cat key-11
key-11         LCB_KEY_ENOENT (0x0D)
```

In the above output, you will see that all of the 10 keys are successfully found. If a key was not found (as in key-11), you would have seen a “LCB_KEY_ENOENT” error message.

This concludes Lab #4.0

In the next lab, we will rejoin Nodes #3 and #4 back into the cluster.