

## 1. Table of Contents - Version:20220130

2.	Prerequisite.....	3
3.	Red Hat and CentOS Installation – 60 Minutes(D) .....	7
4.	Windows Installation .....	19
5.	Couchbase Server Startup and Shutdown & Testing Nodes - 30 Minutes(D) .....	24
6.	Couchbase Web Console – 20 Minutes(D).....	32
7.	CLI Reference – 30 Minutes (D) .....	40
8.	Bucket , Scope and Collections– 60 Minutes (D).....	49
9.	Upgrade - TBD.....	70
10.	Cluster Operations - 120 Minutes (D) .....	82
11.	View Creation – 30 Minutes .....	121
12.	Query Workbench & Index– 35 Minutes.....	129
13.	N1QL : Load data in Bucket, create primary index – 35 Minutes.....	144
14.	N1QL - Select documents in <i>Workbench</i> and the <i>cbq</i> tool – 60 Minutes.....	11
15.	Manipulating data using N1QL DML – 20 Minutes.....	30
16.	N1QL Explain-Querying ranges, ordering results, and explaining query details -30 Minutes.....	39
17.	Tuning Query – (Explain to determine query plan) – 30 Minutes.....	49
18.	XDCR – 90 Minutes (D) .....	59
19.	LDAP Authentication & Security Roles – 90 Minutes(DP).....	90
20.	Auditing for Administrators -20 Minutes(D) .....	117
21.	Backup and Restore – 45 Minutes (D) .....	121
22.	Full Text Search – 60 Minutes (D) .....	135
23.	Eventing service – 60 Minutes(D).....	147

24.	Analytics service(s) - 60 Minutes(D) .....	170
25.	Monitoring and Troubleshooting – 60 Minutes.....	205
26.	Advance Cluster Settings – 30 Minutes.....	220
27.	Request profiling & Query Monitoring (Optional) – 90 Minutes.....	230
28.	Advance command:.....	275
	Mount Shared Folder .....	277
1.	OpenLdap Installation.....	277
	Linux.....	277
	Windows.....	283

Software : Couchbase 7.0.2

## 2. Prerequisite

### Using VM:

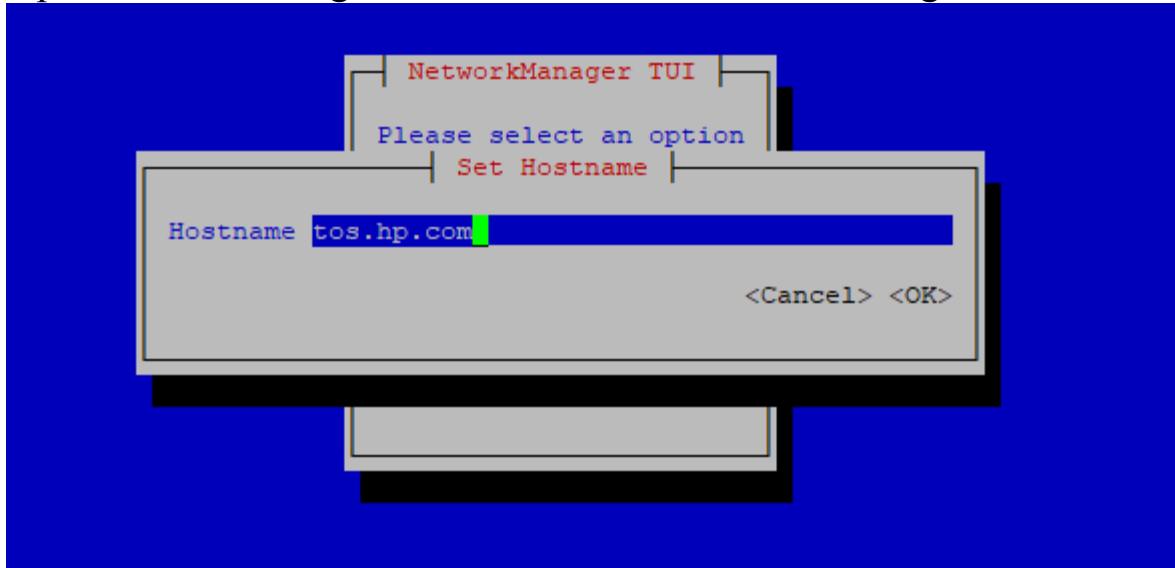
Refer the Prerequisite file for starting the VM.

Verify the host name and ensure to set as shown below:

tos.hp.com

Steps to set the hostname:

Open the GUI using nmtui and set the Hostname using the console.

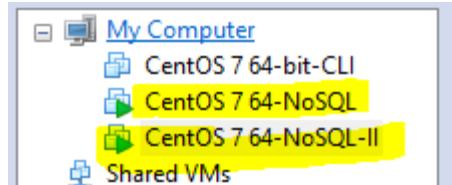


Save the setting.

### 2 VM

You need to have two VM machines to perform this lab. Hence copy the earlier VM machine and rename as shown below. You need to shut down the earlier VM before copying the VM folder. Moreover, you should shut down the earlier machine before initiating copy.

You need to have two VM folders as shown above. Then import the second VM on the workstation



Now start both the VMs.



### Centos 6.0

vi /etc/sysconfig/network

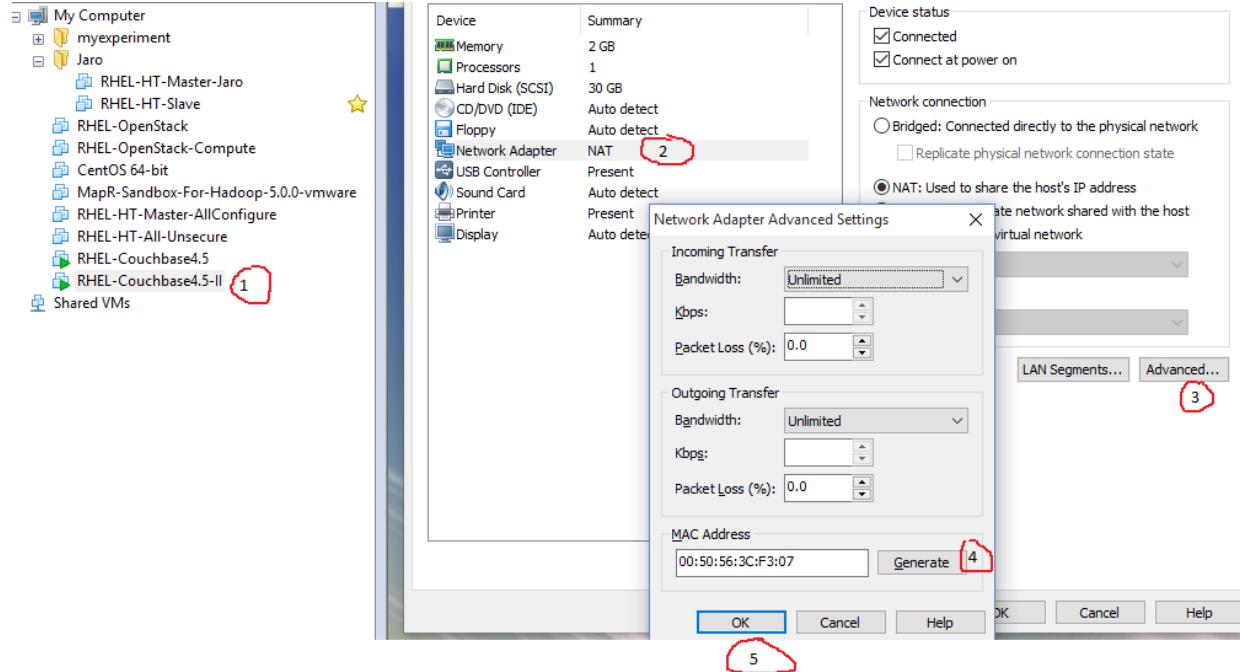
vi /etc/hosts

### centos 7.0

Shutdown the slave machine and regenerate the MAC address as shown below

Ensure to regenerate the mac address for the second machine .i.e slave

Right click on the Second VM → Setting → Select the Network Adapter → Advance → Generate. E.x as shown below



Reboot the slave machine.[Note: reboot]

Slave

You can use nmtui command to change the hostname in centos 7.

## Using Docker

Install docker in your host. Refer any documentation on web.

Start docker and perform the following:

Define a network and create a container attaching to the network.

```
# docker network create --driver bridge spark-net
```

```
# docker run -it --name couchbaseo --hostname couchbaseo.master.com --privileged -p 8091-8094:8091-8094 -p 11210:11210 --network spark-net -v
```

```
/Users/henrypotsangbam/Documents/Docker:/opt centos:7 /usr/sbin/init
```

```
(base) Henrys-MacBook-Air:~ henrypotsangbam$ docker run --name couchbase0 -p 8091:8091 -i -t ehenry0227/learning:spark-kafka-raw
Unable to find image 'ehenry0227/learning:spark-kafka-raw' locally
spark-kafka-raw: Pulling from ehenry0227/learning
75f829a71a1c: Already exists
ef144c6a9e96: Pull complete
149fc595223e: Pull complete
Digest: sha256:db03e74077238ab0c3242cf97beb1f5b8dd240e0da19995e9e2604a0b68f820b
Status: Downloaded newer image for ehenry0227/learning:spark-kafka-raw
[root@e0d66b6bb3c9 ~]# docker images
```

```
#yum install bzip2
```

2 Node

```
# docker run -it --name couchbase1 --hostname couchbase1.master.com --privileged -p 9091:8091 -p 9094:8094 -p 21210:11210 --network spark-net -v
```

```
/Users/henrypotsangbam/Documents/Docker:/opt1 centos:7 /usr/sbin/init
```

### 3. Red Hat and CentOS Installation – 60 Minutes(D)

// For all command started with #/\$ need to be executed in a command terminal  
Open SSL is required for the Couchbase installation.

For CentOS systems, you can query the version of the OpenSSL package installed on your system by typing:

```
# rpm -q -a | grep "openssl"
```

You should receive output that looks like this:

```
openssl-1.0.1e-16.el6_5.7.x86_64
```

```
[root@master ~]# cd /couchbase/  
[root@master couchbase]# ls  
[root@master couchbase]# ls  
couchbase-server-enterprise-4.5.0-centos6.x86_64.rpm  
[root@master couchbase]# rpm -a -q | grep openssl  
openssl-1.0.0-27.el6.x86_64  
[root@master couchbase]#
```

If you are on one of the supported distributions, ensure that your OpenSSL version is up-to-date.

#### Complete the Installation

Software : couchbase-server-enterprise-7.0.0-beta-centos7.x86\_64.rpm

You must be logged in as root (superuser) or use the sudo command to complete the installation. Use the following command to install the Couchbase Server package:

Ensure to copy the software to the /Software folder

```
# cd /Software  
# rpm --install couchbase-server-enterprise-* .rpm
```

After the rpm command completes, the Couchbase Server service starts automatically. It is configured to start automatically under OS runlevels 2, 3, 4, and 5.

After installation is completed, the installation process displays a message similar to the following:

```
[root@tos Software]# rpm --install couchbase-server-enterprise-6.0.1-centos7.x86_64.rpm  
Warning: Transparent hugepages looks to be active and should not be.  
Please look at https://developer.couchbase.com/documentation/server/current/install/thp-disable.html as for how to PERMANENTLY alter th  
is setting.  
Warning: Swappiness is not set to 0.  
Please look at https://developer.couchbase.com/documentation/server/current/install/install-swap-space.html as for how to PERMANENTLY a  
lter this setting.  
Minimum RAM required : 4 GB  
System RAM configured : 1.79 GB  
  
Minimum number of processors required : 4 cores  
Number of processors on the system : 2 cores  
  
Created symlink from /etc/systemd/system/multi-user.target.wants/couchbase-server.service to /usr/lib/systemd/system/couchbase-server.s  
ervice.  
  
You have successfully installed Couchbase Server.  
Please browse to http://master:8091/ to configure your server.  
Please refer to http://couchbase.com for additional resources.  
  
Please note that you have to update your firewall configuration to  
allow connections to the following ports:  
4369, 8091 to 8094, 9100 to 9105, 9998, 9999, 11207, 11209 to 11211,  
11214, 11215, 18091 to 18093, and from 21100 to 21299.  
  
By using this software you agree to the End User License Agreement.  
See /opt/couchbase/LICENSE.txt.  
  
[root@tos Software]#
```

After the installation is completed, use the service command to manage the Couchbase Server service, including checking the current status. .

Stop the firewall:

```
service iptables stop / systemctl stop firewalld
```

```
[root@localhost couchbase]# service iptables stop
iptables: Flushing firewall rules: [ OK ]
iptables: Setting chains to policy ACCEPT: filter [ OK ]
iptables: Unloading modules: [ OK ]
[root@localhost couchbase]#
```

And disable it:

```
systemctl disable firewalld
```

```
[root@tos Software]# systemctl stop firewalld
[root@tos Software]# systemctl disable firewalld
Removed symlink /etc/systemd/system/dbus-org.fedoraproject.FirewallD1.service.
Removed symlink /etc/systemd/system/basic.target.wants/firewalld.service.
[root@tos Software]#
```

To perform the initial Couchbase Server setup, open a web browser and access the Couchbase Web Console.

After completing the installation, your next step is to initialize the cluster. You can initialize the cluster using the Couchbase Web Console, CLI, or REST API. Only Full Administrators can perform the initial Couchbase Server setup.

<http://tos.hp.com:8091/>

Once you have connected, the **Welcome** screen appears:



The **Welcome** screen lets you either **Setup New Cluster**, or **Join Existing Cluster**. Information on joining an existing cluster is provided below, in the section *Join an Existing Cluster*. To set up a new cluster, left-click on **Setup New Cluster**.

Set Up a New Cluster

The **New Cluster** screen now appears, as follows:

The screenshot shows the 'New Cluster' setup screen in the Couchbase interface. The 'Cluster Name' field contains 'Tos.Couchbase'. The 'Create Admin Username' field contains 'Administrator'. The 'Create Password' and 'Confirm Password' fields both show redacted content. At the bottom left is a 'Back' button, and at the bottom right is a blue 'Next: Accept Terms' button.

Enter the following details:

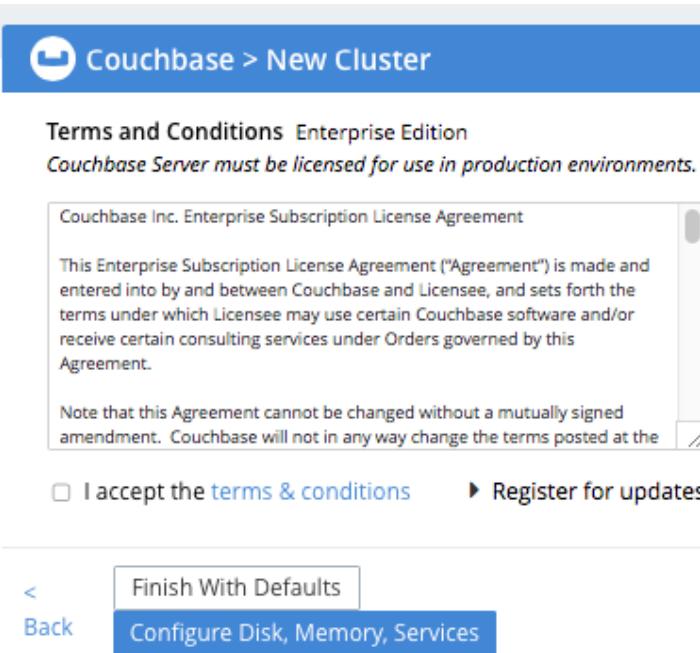
Cluster Name: Tos.Couchbase

Password : admin123

When you have entered appropriate data into each field, left-click on the **Next: Accept Terms** button, at the lower right

Accept Terms

The **New Cluster** screen now changes, to show the **Terms and Conditions** for the Enterprise Edition of Couchbase Server:



Check the **I accept the terms & conditions** checkbox.

To customize those settings, left-click on the **Configure Disk, Memory, Services** button, and proceed as follows.

### Configure Couchbase Server

The **Configure** screen now appears, as follows: (Ensure to update the hostname as tos.hp.com)

 Couchbase > New Cluster > Configure

**Host Name / IP Address** Fully-qualified domain name  
couchbase0.master.com

enable node-to-node encryption ⓘ

**IP Family Preference ⓘ**  
 IPv4  IPv6  IPv4-only  IPv6-only

**Service Memory Quotas ⓘ** Per service / per node

<input checked="" type="checkbox"/> Data	512	MiB
<input checked="" type="checkbox"/> Query	-----	
<input checked="" type="checkbox"/> Index	512	MiB
<input checked="" type="checkbox"/> Search	256	MiB
<input checked="" type="checkbox"/> Analytics	1024	MiB
<input checked="" type="checkbox"/> Eventing	256	MiB
<input checked="" type="checkbox"/> Backup	-----	

TOTAL QUOTA 2560MiB

[< Back](#) Save & Finish

 Couchbase > New Cluster > Configure

RAM Available 4942MiB Max Allowed Quota 3953MiB

**Index Storage Setting**

Standard Global Secondary  
 Memory-Optimized ⓘ

**Data Disk Path** Path cannot be changed after setup  
/opt/couchbase/var/lib/couchbase/data  
Free: 11 GiB

**Indexes Disk Path** Used by GSI, FTS, and Views  
/opt/couchbase/var/lib/couchbase/data  
Free: 11 GiB

**Eventing Disk Path** Path cannot be changed after setup  
/opt/couchbase/var/lib/couchbase/data  
Free: 11 GiB

**Analytics Disk Paths** Paths cannot be changed after setup  
/opt/couchbase/var/lib/couchbase/data  
Free: 11 GiB + -

**Java Runtime Path** optional  
[Input field]

[< Back](#) Save & Finish

The total memory quota you have allocated is displayed below these fields, towards the right. The total RAM available is displayed below this figure, at the center. If your memory allocation is excessive, a notification warns you, and you must lessen your allocation.

When you have finished entering your configuration-details, left-click on the **Save & Finish** button, at the lower right. This configures the server accordingly, and brings up the Couchbase Web Console **Dashboard**, for the first time.

tos.couchbase > Dashboard

Enterprise Edition 7.0.3 build 7031 - IPv4 © 2021 Couchbase, Inc.

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

- Dashboard
- Servers
- Buckets
- Backup
- XDCR
- Security
- Settings
- Logs

- Documents
- Query
- Indexes
- Search
- Analytics
- Eventing
- Views

If this is the first server in the cluster, a notification appears, stating that no buckets are currently defined. A *bucket* is the principal unit of data-storage used by Couchbase Server. In order to save and subsequently access documents and other objects, you must create one or more buckets.

As specified by the notification, you can go to **Buckets**, and begin bucket-creation::

Click on Add Bucket, to create a default Bucket.

The screenshot shows the Tos.Couchbase web interface. The URL in the address bar is `tos.master.com:8091/ui/index.html#!/buckets`. The top navigation bar includes links for Activity, Documentation, Support, and Administrator. The main header 'Tos.Couchbase > Buckets' is displayed. A large yellow button labeled 'ADD BUCKET' is visible. A message box in the center says 'You have no data buckets. Use "ADD BUCKET" above to create one, or load a sample bucket with data & indexes.' On the left, a sidebar has tabs for Dashboard, Servers, Buckets (which is highlighted in blue), and Indexes.

Enter the following details. Then click Add bucket

**Add Data Bucket**

**Name**  
default

**Memory Quota** in megabytes per server node  
100 MB

███████████  
■ other buckets (0 B)   ■ this bucket (100 MB)   ■ remaining (512 MB)

**Bucket Type**  
 Couchbase    Memcached    Ephemeral

► Advanced bucket settings

[Cancel](#) [Add Bucket](#)

You should be able to see the bucket which you have created just now as shown below:

Tos.Couchbase > Buckets

name ▾	items	resident	ops/sec	RAM used/quota	disk used
default	0	100%	0	20.5MB / 100MB	34B

[Documents](#) [Statistics](#)

Click on the Server's option on the left menu to determine the number of node present in the couchbase cluster. Currently we should have only one node as shown below:

tos.couchbase > Servers

Dashboard	filter servers...	Rebalance					
Servers	group	services	CPU	RAM	swap	disk used	items
couchbase0.master.com	Group 1	data query index search analytics eventing backup	0.0%	---	---	---	0/0

[Statistics](#)

Done

**Couchbase Server is now running and ready to use.**

-----Completed Installation Lab -----

## 4. Windows Installation

To install Couchbase Server on Windows using interactive install wizard follow the steps below:

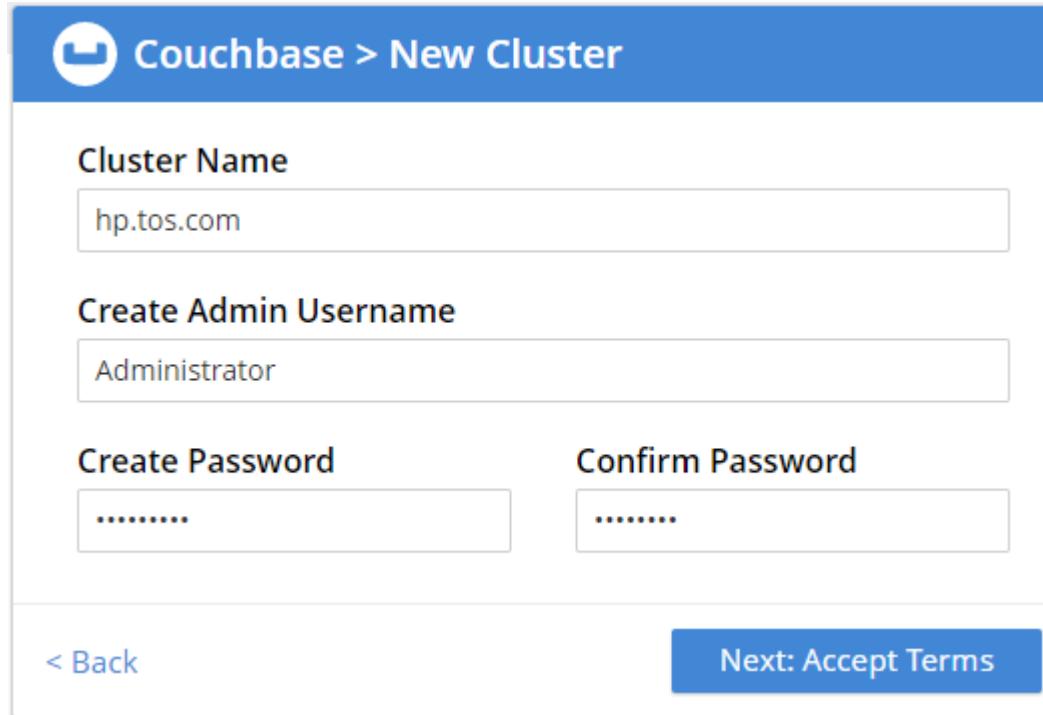
1. In Windows Explorer, locate the downloaded `couchbase-server-enterprise_version-windows_amd64.msi` file, which typically is located in the default Downloads folder. Double click on the executable file.
2. You will be prompted with the Installation Location screen. You can change the location where the Couchbase Server application is located, which configures the server location and not the location where the persistent data is stored.
3. Click Yes to continue.
4. After installation, follow the instructions to [Create a Cluster](#).

Access the URL <http://localhost:8091>

Once you have connected, the Welcome screen appears:



The Welcome screen lets you either Setup New Cluster, or Join Existing Cluster. Information on joining an existing cluster is provided in [Join a Cluster and Rebalance](#). To set up a new cluster, left-click on Setup New Cluster.



The screenshot shows the 'New Cluster' setup screen for Couchbase. At the top, there's a blue header bar with the Couchbase logo and the text 'Couchbase > New Cluster'. Below the header, there are four input fields: 'Cluster Name' containing 'hp.tos.com', 'Create Admin Username' containing 'Administrator', 'Create Password' (redacted), and 'Confirm Password' (redacted). At the bottom left is a 'Back' button labeled '< Back', and at the bottom right is a blue 'Next: Accept Terms' button.

Cluster Name  
hp.tos.com

Create Admin Username  
Administrator

Create Password  
.....

Confirm Password  
.....

< Back      Next: Accept Terms

Click Customize:

 Couchbase > New Cluster / Configure

**Host Name / IP Address** Usually localhost or similar  
hp.tos.com  
Requested name hostname is not allowed: short names are not allowed.  
Couchbase Server requires at least one dot in a name

**Data Disk Path** Path cannot be changed after setup  
d:/MyExperiment/Couchbase/var/lib/couchbase/data  
Free: 370 GB

**Indexes Disk Path** Path cannot be changed after setup  
d:/MyExperiment/Couchbase/var/lib/couchbase/data  
Free: 370 GB

**Couchbase Memory Quotas** Per service / per node

<input checked="" type="checkbox"/> Data Service	1024	MB
<input checked="" type="checkbox"/> Index Service	256	MB
<input checked="" type="checkbox"/> Search Service	256	MB
<input checked="" type="checkbox"/> Query Service	-----	

TOTAL QUOTA 1536MB

RAM Available 8106MB Max Allowed Quota 7082MB

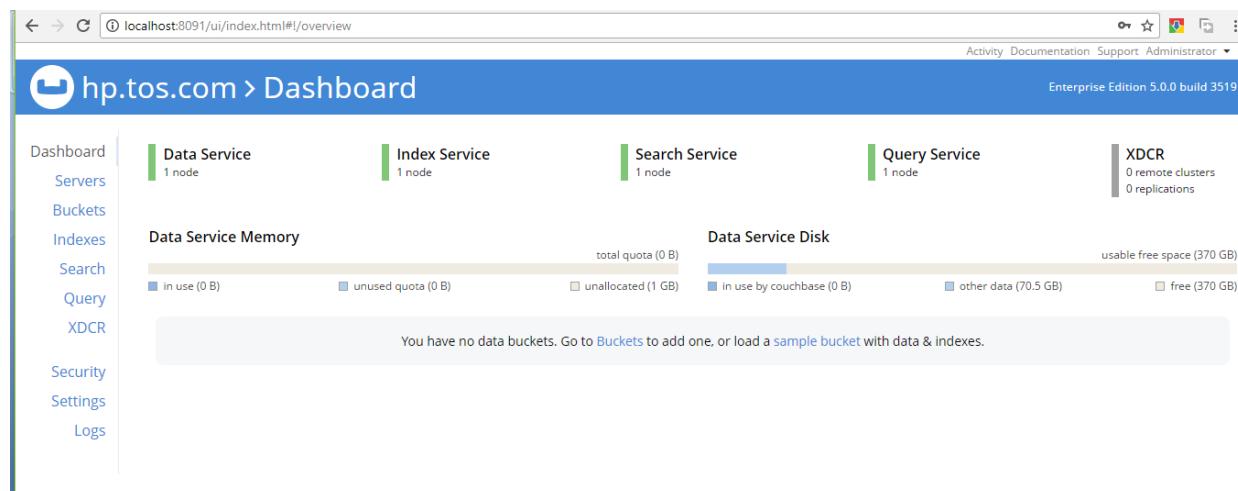
### Index Storage Setting

Standard Global Secondary

Memory-Optimized (i)

Enable software update notifications in the web console

[< Back](#) Save & Finish



## 5. Couchbase Server Startup and Shutdown & Testing Nodes - 30 Minutes(D)

Use startup and shutdown scripts to manually start or shut down Couchbase Server.

### Start and Stop on Linux

On Linux, Couchbase Server is installed as a standalone application with support for running as a background (daemon) process during start-up through the use of a standard control script, which is located in/etc/init.d/couchbase-server.

The startup script is automatically installed during installation from one of the Linux packaged releases (Debian/Ubuntu or Red Hat/CentOS). By default, Couchbase Server is configured to be started automatically at run levels 2, 3, 4, and 5, and explicitly shut down at run levels 0, 1 and 6.

Verify the status of the Couchbase server:

```
#systemctl status couchbase-server or service couchbase-server status
```

```
[root@tos ~]# systemctl status couchbase-server
● couchbase-server.service - Couchbase Server
   Loaded: loaded (/usr/lib/systemd/system/couchbase-server.service; enabled; vendor preset: disabled)
     Active: active (running) since Wed 2018-01-17 23:26:42 IST; 28min ago
       Docs: http://docs.couchbase.com
 Main PID: 2602 (beam.smp)
    CGroup: /system.slice/couchbase-server.service
            └─2602 /opt/couchbase/lib/erlang/erts-5.10.4.0.0.1/bin/beam.smp -A...
              ├─2617 /opt/couchbase/lib/erlang/erts-5.10.4.0.0.1/bin/epmd -daemo...
              ├─2667 /opt/couchbase/bin/gosecrets
              ├─2671 /opt/couchbase/lib/erlang/erts-5.10.4.0.0.1/bin/beam.smp -A...
              ├─2697 sh -s disksup
              ├─2698 /opt/couchbase/lib/erlang/lib/os_mon-2.2.14/priv/bin/memsup...
              ├─2700 /opt/couchbase/lib/erlang/lib/os_mon-2.2.14/priv/bin/cpu_su...
              ├─2701 inet_gethost 4
              ├─2702 inet_gethost 4
              ├─2769 /opt/couchbase/bin/saslauthd-port
```

```
              └─2958 /opt/couchbase/bin/projector -kvaddrs=127.0.0.1:11210 -admi...
              └─2967 /opt/couchbase/bin/goport -graceful-shutdown=false -window...
              └─2971 /opt/couchbase/bin/indexer -vbuckets=1024 -cluster=127.0.0....
              └─2982 /opt/couchbase/bin/goport -graceful-shutdown=false -window...
              └─2986 /opt/couchbase/bin/cbq-engine --datastore=http://127.0.0.1:...
              └─2996 /opt/couchbase/bin/goport -graceful-shutdown=false -window...
              └─3000 /opt/couchbase/lib/erlang/erts-5.10.4.0.0.1/bin/beam.smp -P...
              └─3001 /opt/couchbase/bin/cbft -cfg=metakv -uuid=0cffce5a5109282ce...

Jan 17 23:26:42 tos.master.com systemd[1]: Started Couchbase Server.
Jan 17 23:26:42 tos.master.com systemd[1]: Starting Couchbase Server...
Jan 17 23:26:43 tos.master.com couchbase[2602]: {error_logger,{{2018,1,17},{...}}}
Hint: Some lines were ellipsized, use -l to show in full.
```

The above command shows that the couchbase is running.

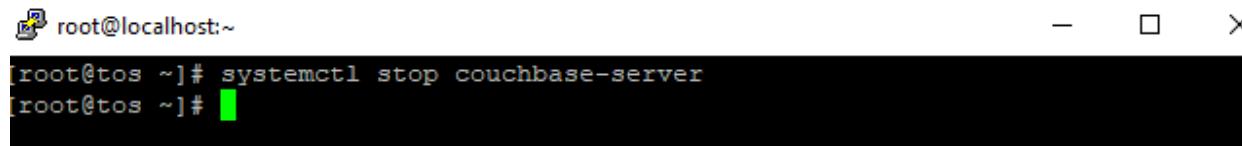
To manually stop Couchbase Server using the startup/shutdown script:

- For RHEL 6, as a root user:

```
# sudo service couchbase-server stop
```

## For RHEL 7:

```
# systemctl stop couchbase-server
```



The screenshot shows a terminal window with a black background and white text. The title bar says 'root@localhost:~'. The command 'systemctl stop couchbase-server' is typed in, followed by a blank line where the output would appear.

```
[root@tos ~]# systemctl stop couchbase-server
[root@tos ~]#
```

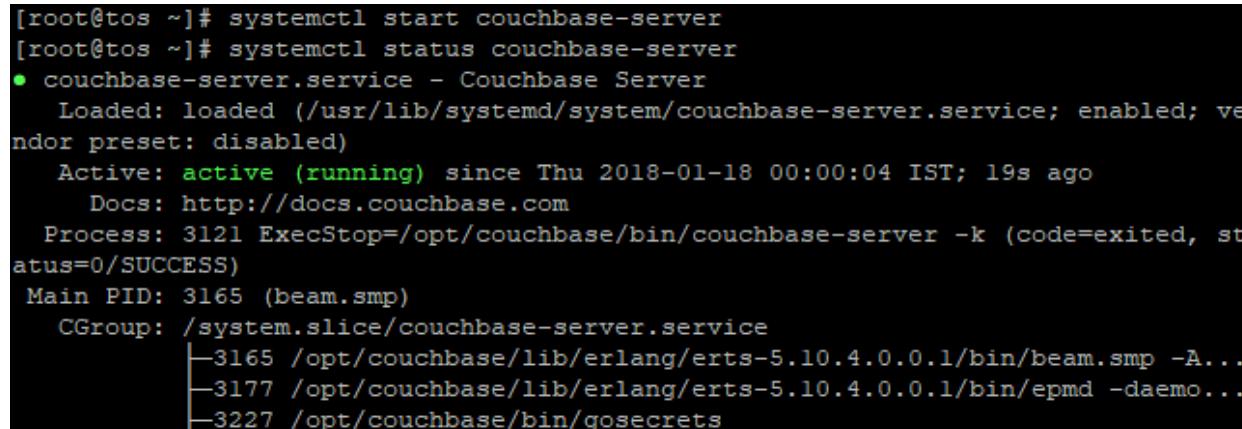
To manually start Couchbase Server using the startup/shutdown script:

- For RHEL 6:

```
# service couchbase-server start
```

## For RHEL 7:

```
# sudo systemctl start couchbase-server
```



The screenshot shows a terminal window with a black background and white text. It displays the output of the 'systemctl start couchbase-server' command followed by the 'systemctl status couchbase-server' command. The status output shows the service is active and running.

```
[root@tos ~]# systemctl start couchbase-server
[root@tos ~]# systemctl status couchbase-server
● couchbase-server.service - Couchbase Server
   Loaded: loaded (/usr/lib/systemd/system/couchbase-server.service; enabled; vendor preset: disabled)
     Active: active (running) since Thu 2018-01-18 00:00:04 IST; 19s ago
       Docs: http://docs.couchbase.com
     Process: 3121 ExecStop=/opt/couchbase/bin/couchbase-server -k (code=exited, status=0/SUCCESS)
    Main PID: 3165 (beam.smp)
      CGroup: /system.slice/couchbase-server.service
              └─3165 /opt/couchbase/lib/erlang/erts-5.10.4.0.0.1/bin/beam.smp -A...
                ├─3177 /opt/couchbase/lib/erlang/erts-5.10.4.0.0.1/bin/epmd -daemo...
                ├─3227 /opt/couchbase/bin/gosecrets
```

You can verify it as shown above after starting the node.

## On Windows

On Windows, Couchbase Server is installed as a Windows service. You can use the Services tab within the Windows Task Manager to start and stop Couchbase Server.

You will need a power user or administrator privileges, or have separately granted rights to manage services to start and stop Couchbase Server. By default, the service automatically starts when the machine boots.

Couchbase Server can be started and stopped via Windows Task Manager, Windows system net command, and Couchbase-supplied .bat scripts.

### Start manually with the Task Manager

To manually start the service from the Windows interface:

1. Open the Windows Task Manager and select the Services tab to open the Services management console.
2. Alternatively, select the Start, select Run and then type Services.msc to open the Services management console.
3. Locate the Couchbase Server service and right-click.
4. Select **Start** or **Stop** as appropriate.

**Note:** You can also alter the configuration so that the service is not automatically started during boot.

### Start and Stop with the net command

To start and stop Couchbase Server using net:

```
net start CouchbaseServer
```

```
net stop CouchbaseServer
```

### **Start and stop with the .bat scripts**

The Couchbase-supplied start and stop scripts are provided in the standard installation in the bin directory.

To start and stop Couchbase Server, use the scripts located in:

C:\Program Files\Couchbase\Server\bin\service\_start.bat

C:\Program Files\Couchbase\Server\bin\service\_stop.bat

### **Testing Couchbase Server**

Testing the connection to the Couchbase Server can be performed in a number of different ways.

To verify that your installation works for clients, you can use either the cbworkloadgen command, or telnet. The cbworkloadgen command uses the Python Client SDK to communicate with the cluster, checking both the cluster administration port and data update ports.

### **Testing with cbworkloadgen**

The command cbworkloadgen executes a number of different operations to provide basic testing functionality for Couchbase Server. It does not provide performance or workload testing.

To test a Couchbase Server installation using the command `cbworkloadgen`, execute the command supplying the IP address of the running node:

```
#cd /opt/couchbase/bin
```

```
./cbworkloadgen -n localhost:8091 -u Administrator -p admin123
```

```
[root@tos bin]# ./cbworkloadgen -n localhost:8091 -u Administrator -p admin123
[########################################] 100.0% (10527/estimated 10526 msgs)
bucket: default, msgs transferred...
      :          total |     last |   per sec
byte  :           105270 |    105270 |  210292.7
done
[root@tos bin]#
```

The progress and activity of the tool can also be monitored within the Couchbase Web Console.

For a longer test you can increase the number of iterations:

```
./cbworkloadgen -n localhost:8091 -u Administrator -p admin123 --max-items=100000
```

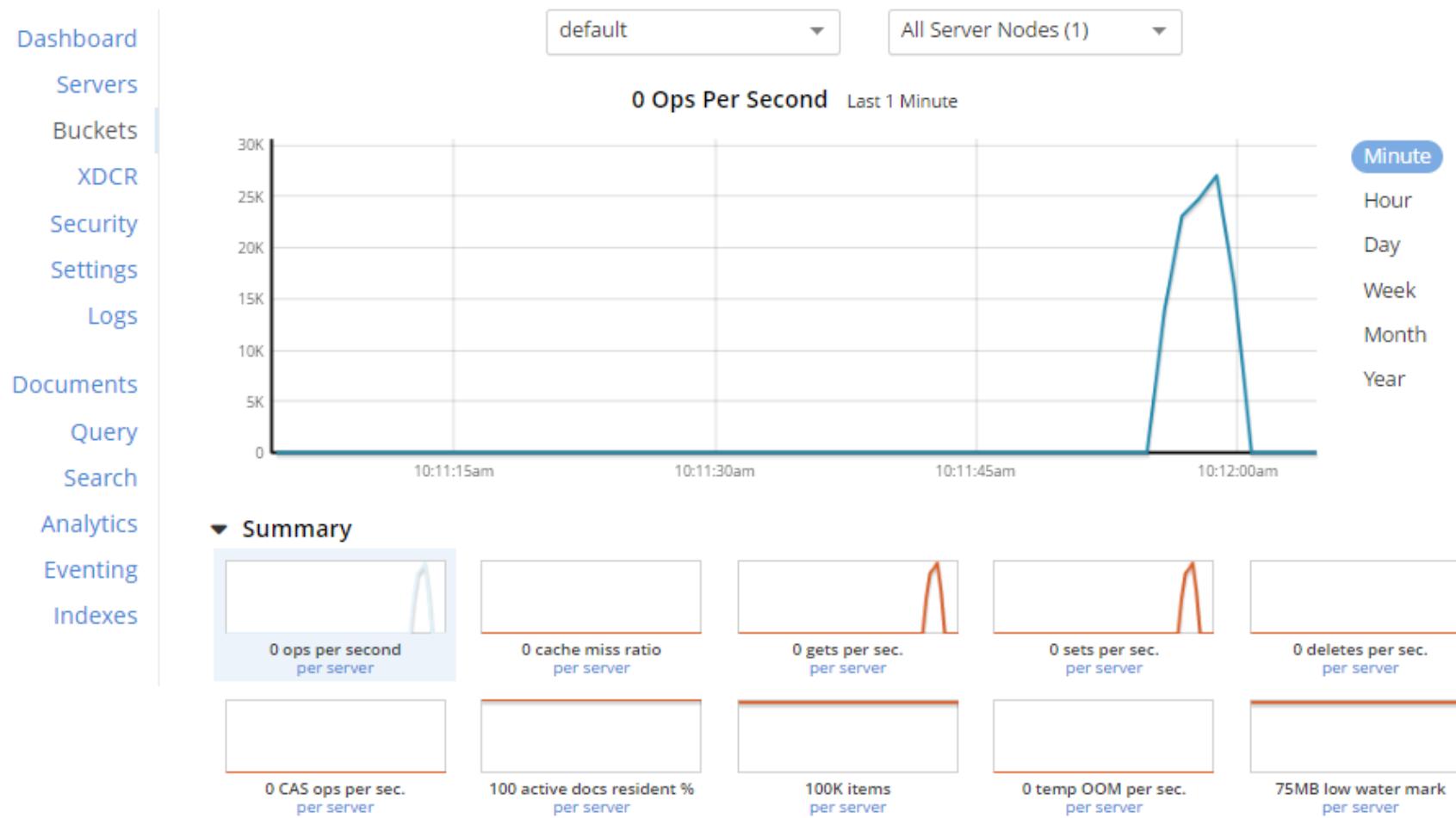
```
done
[root@tos bin]# ./cbworkloadgen -n localhost:8091 -u Administrator -p admin123
--max-items=100000
[########################################] 100.0% (105264/estimated 105263 msgs)
bucket: default, msgs transferred...
      :          total |     last |   per sec
byte  :           1052640 |    1052640 |  172255.2
done
[root@tos bin]#
```

The screenshot shows the Couchbase Admin UI with the following details:

- Top Bar:** Activity, Documentation, Support, Administrator ▾
- Breadcrumbs:** Tos.Couchbase > Buckets
- Add Bucket:** ADD BUCKET
- Left Sidebar:** Dashboard, Servers, Buckets (selected), Indexes, Search, Query, XDCR, Security, Settings.
- Buckets Table:**

	name ▾	items	resident	ops/sec	RAM used/quota	disk used	Documents	Statistics
	default	100,000	100%	0	20.5MB / 100MB	18.9MB	<a href="#">Documents</a>	<a href="#">Statistics</a>

Bucket → Default → Statistics or Servers -> Statistics  
You can verify the Ops per seconds.



----- End of Lab -----

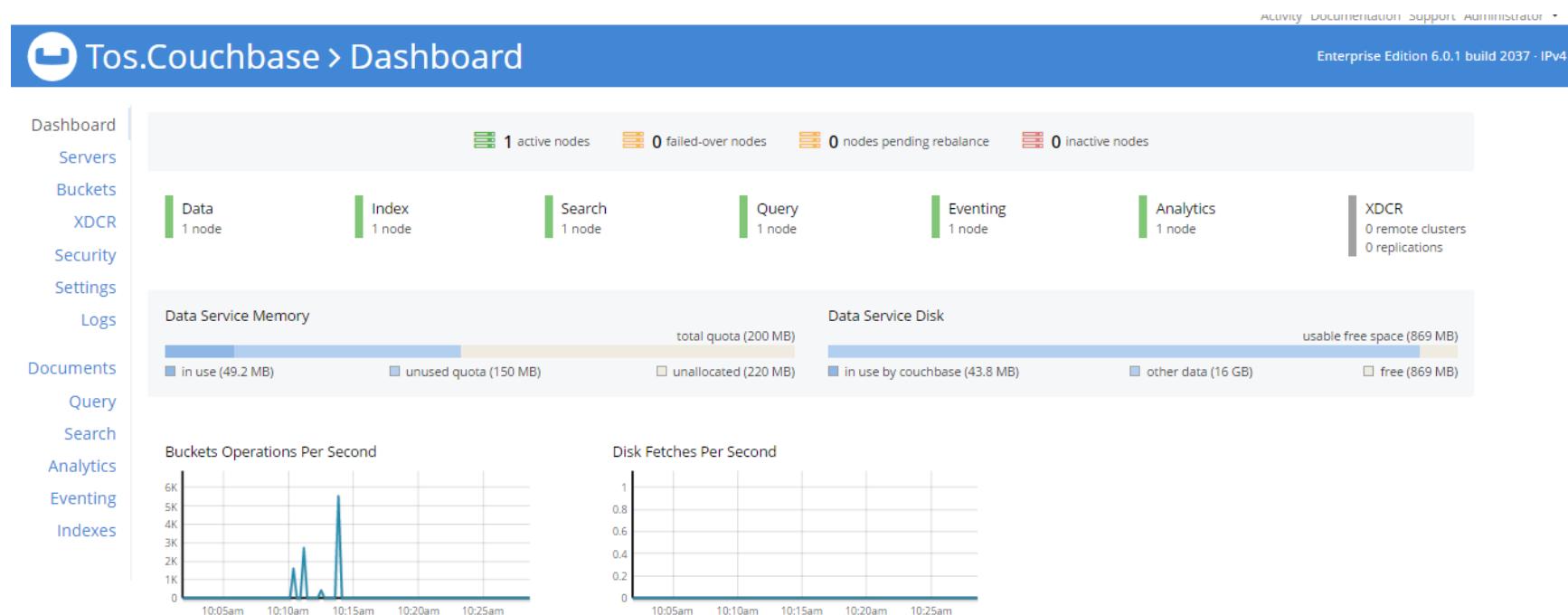
## 6. Couchbase Web Console – 20 Minutes(D)

Start couchbase server. In this lab we will familiarize some of the options of the Admin Web console. We will be going in details as we moved on. In this lab we will configure bucket and create some documents in it. Access the Admin console using the following URL:

<http://tos.master.com:8091/index.html>

Credentials: - Administrator/root123

Once you logon from the web console, you will get the dashboard as shown below.



The Couchbase Web Console is the main tool for managing the Couchbase environment.

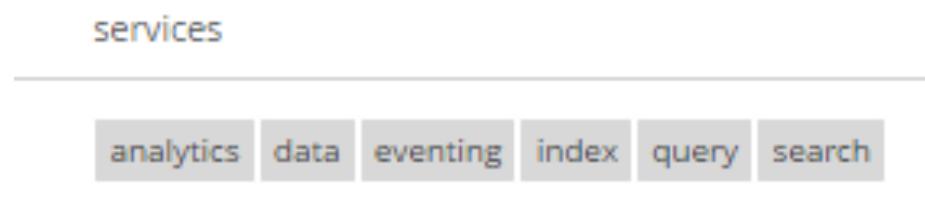
When you start the Couchbase Web Console, the Dashboard screen opens with the tab Overview of the whole cluster selected by default.

## Servers Overview

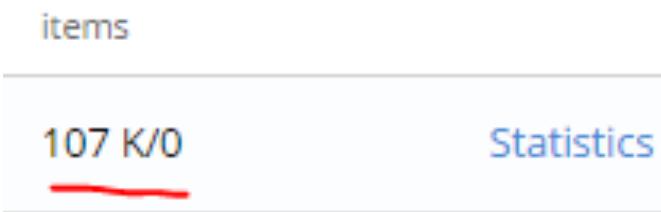
It provides information of the services configured in that particular cluster and what services are running in the individual nodes. For example in this screen shot, data services are configured in 1 node. You can click on the 1 node link in the Data service to get the details of the nodes.

								<a href="#">Failover Multiple Nodes</a>	<a href="#">Rebalance</a>	
name ▾	group	services	CPU	RAM	swap	disk used	items			
192.168.139.129	Group 1	<a href="#">analytics</a> <a href="#">data</a> <a href="#">eventing</a> <a href="#">index</a> <a href="#">query</a> <a href="#">search</a>	8.8%	68.8%	0%	43.8MB	107 K/0	<a href="#">Statistics</a>		

In the above, it shows the one node which have data services configured and provide an overview of the system resources usages. You can monitor the overall health of the node using the mention option. The following services are running in the mention node.



You can also determine whether data are distributed equally among the nodes using this console. Item should be equally distributed on the various nodes when the couchbase cluster have multiple nodes.



## Setting

This section specify how much RAM is allocated for the cluster and how much is being consumed.

**Cluster Name**

tos.couchbase

**Memory Quotas** per server node

Note: Total Couchbase RAM allocation for each node cannot exceed 90% of its available memory.

**Data**

512 MiB

**Query**

-----

**Index**

512 MiB

**Search**

256 MiB

**Analytics**

1024 MiB

**Eventing**

256 MiB

**Backup**

-----

**Current Version***Couchbase Server Enterprise Edition 7.0.3 build 7031* Share usage information with Couchbase and get software update notifications.**Save**

## Server --> Server Nodes

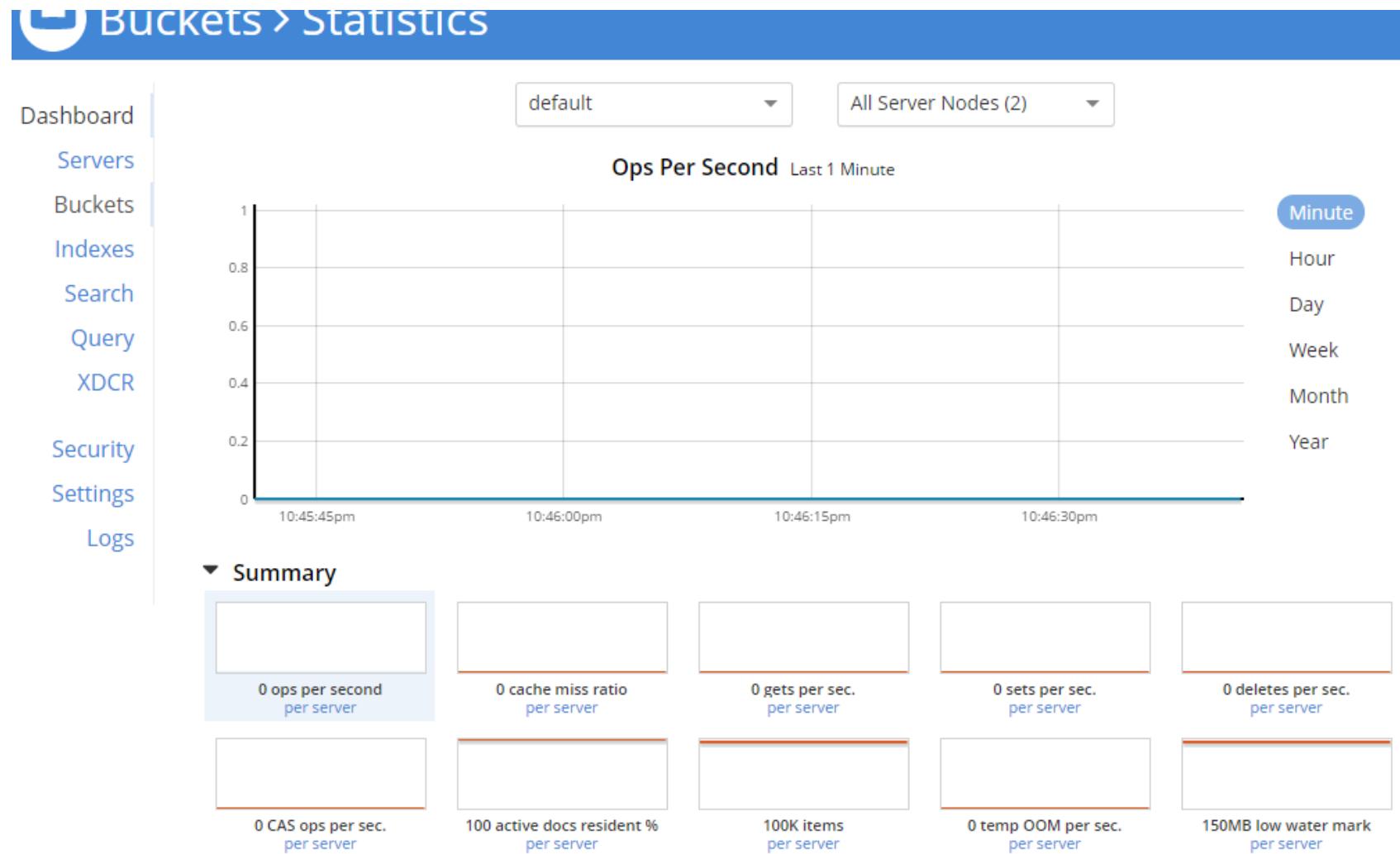
The Server Nodes monitoring overview shows summary data for the Swap Usage, RAM Usage, CPU Usage and Active Items across all the nodes in your cluster

Dashboard								<a href="#">Failover Multiple Nodes</a>	<a href="#">Rebalance</a>
Servers	name ▾	group	services	CPU	RAM	swap	disk used	items	
	192.168.139.129	Group 1	<a href="#">analytics</a> <a href="#">data</a> <a href="#">eventing</a> <a href="#">index</a> <a href="#">query</a> <a href="#">search</a>	4.81%	68.2%	0%	43.8MB	107 K/0	<a href="#">Statistics</a>

Click on the row next to a server displays server node specific information, including the IP address, OS, Couchbase version and Memory and Disk allocation information.

name ▾	group	services	CPU	RAM	swap	disk used	items	
<a href="#">tos.master.com</a>	Group 1	<a href="#">data</a> <a href="#">full text</a> <a href="#">index</a> <a href="#">query</a>	35.9%	84%	1.05%	135MB	69.4 K/69.4 K	<a href="#">Statistics</a>
Uptime: 1 hour, 26 minutes, 3 seconds OS: x86_64-unknown-linux-gnu Version: Enterprise Edition 5.0.1 build 5003 Data Service RAM Quota: 512 MB Data Storage Path: /opt/couchbase/var/lib/couchbase/data Index Storage Path: /opt/couchbase/var/lib/couchbase/data		<b>Memory</b> 	<b>Disk Storage</b> 					

How Much Ram is allocated for each node? Go through the above Diagram.  
 Click on Servers -> Statistics -> Data Buckets to get details about it-> Default → Statistics



Load the sample bucket : beer-sample.

Using the webconsole : Settings → Sample Buckets → select the beer-sample bucket which is on the Available Samples column

Sample buckets contain example data and Couchbase views.  
You can provision one or more sample buckets to help you discover the power of Couchbase Server.  
Sample buckets (like all buckets in Couchbase Server 5.0+) can only be accessed by a user with privileges for that bucket.

Available Samples	Installed Samples
<input checked="" type="checkbox"/> beer-sample	travel-sample
<input type="checkbox"/> gamesim-sample	

**Load Sample Data**

Click on Load Sample Data button to load it.

After a few seconds:

name	items	resident	ops/sec	RAM used/quota	disk used	Documents	Statistics
beer-sample	3,335	100%	552	26MB / 100MB	4.07MB	<a href="#">Documents</a>	<a href="#">Statistics</a>
travel-sample	31,592	100%	0	38.8MB / 100MB	137MB	<a href="#">Documents</a>	<a href="#">Statistics</a>

You will observe that items are loading in the beer-sample bucket as shown above. You can refer the Activity button on the right most top column to determine the status of the loading.  
And you should have 7303 items in the bucket beer-sample

The screenshot shows the Couchbase Web Console interface. On the left, there's a navigation sidebar with links for Dashboard, Servers, Buckets (which is highlighted with a green bar), XDCR, and Security. The main area is titled "Buckets". At the top right, there are buttons for "Activity", "Documentation", "Support", "Administrator", and "ADD BUCKET". Below the title, there's a table with the following columns: name, items, resident, ops/sec, RAM used/quota, and disk used. Two rows are visible: "beer-sample" with 7,303 items (circled in red) and "travel-sample" with 31,592 items.

name	items	resident	ops/sec	RAM used/quota	disk used
beer-sample	7,303	100%	0	27MB / 100MB	25.2MB
travel-sample	31,592	100%	0	38.8MB / 100MB	137MB

----- Lab Ends Here -----

## 7. CLI Reference – 30 Minutes (D)

Most of the CLI command will be in the bin folder:

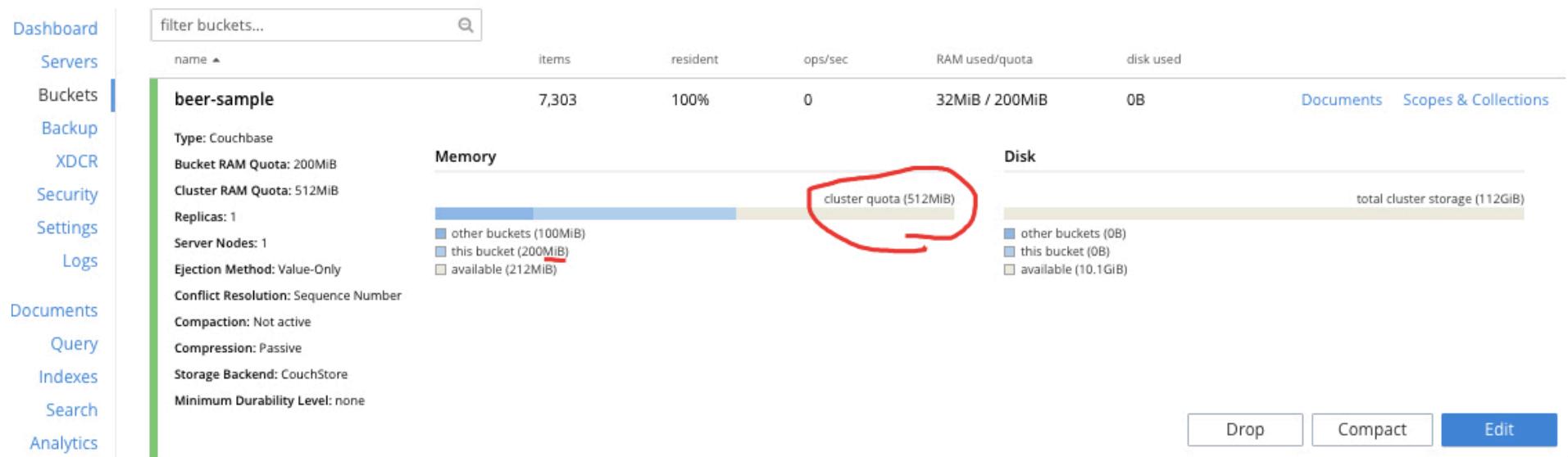
Before executing any command change the directory to it.

```
#cd /opt/couchbase/bin
```

```
[root@localhost sysconfig]# cd /opt/couchbase/bin
[root@localhost bin]# pwd
/opt/couchbase/bin
[root@localhost bin]#
```

### Increase the cluster memory size

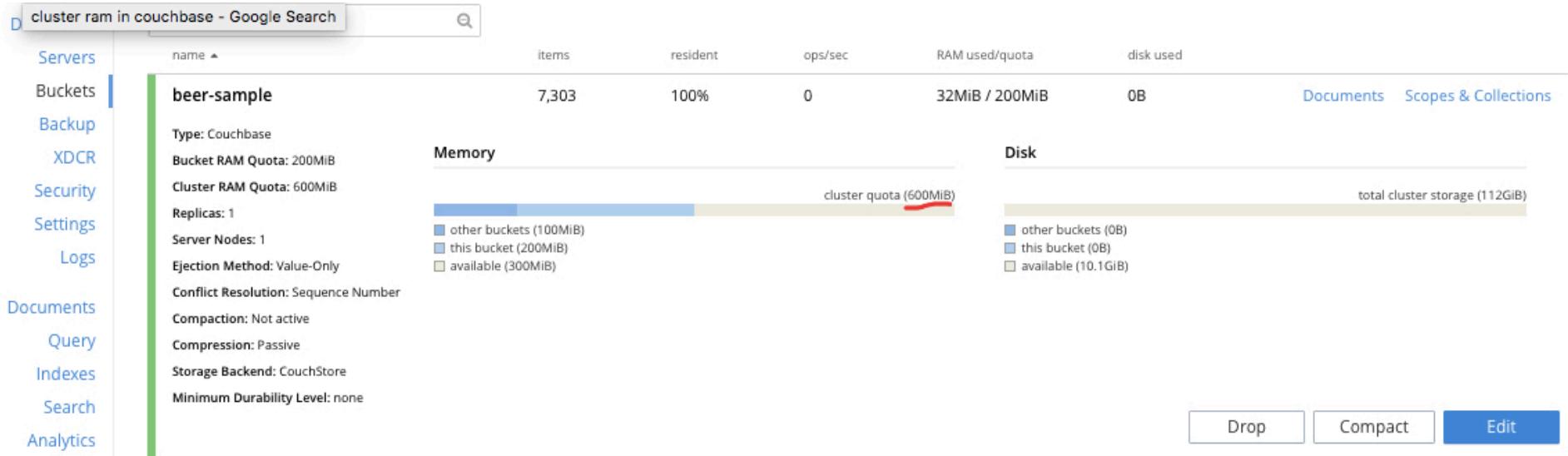
You can note down the current cluster size before increasing it from the web console – Buckets -> beer-sample (Click on it)



Execute the following command to increase the cluster memory size to 612 MB

```
./couchbase-cli setting-cluster -c couchbase0.master.com -u Administrator -p admin123 --cluster-ramsize=600
[root@tos bin]# ./couchbase-cli setting-cluster -c tos.hp.com:8091 -u Administrator -p admin123 --cluster-ramsize=260
SUCCESS: Cluster settings modified
[root@tos bin]#
```

After the successfully execution of the update command, your cluster memory size will be 610 mb



Determine the unallocated/Available size. Ie. Total consume by the cluster subtracted from that of the cluster Memory.

To list all buckets in a cluster:

```
./couchbase-cli bucket-list -c 127.0.0.1:8091 -u Administrator -p admin123
```

```
[root@tos bin]# ./couchbase-cli bucket-list -c 127.0.0.1:8091 -u Administrator -p admin123
beer-sample
bucketType: membase
numReplicas: 1
ramQuota: 104857600
ramUsed: 28411928
travel-sample
bucketType: membase
numReplicas: 1
ramQuota: 104857600
ramUsed: 40726120
[root@tos bin]#
```

Enable the beer-sample bucket before going ahead as shown below: If you already have load the beer-sample bucket , you need to delete it.

Dashboard → Buckets → beer-sample → Delete / Drop( Accept the warning and Proceed).

name	items	resident	ops/sec	RAM used/quota	disk used
<a href="#">beer-sample</a>	7,303	100%	0	27MB / 100MB	25.2MB

Type: Couchbase  
Bucket RAM Quota: 100MB  
Cluster RAM Quota: 260MB  
Replicas: 1  
Server Nodes: 1  
Ejection Method: Value-Only  
Conflict Resolution: Sequence Number  
Compaction: Not active

Memory

Disk

total cluster storage (16.9 GB)

Delete Compact Edit

The sample bucket Zip files can be loaded into the cluster by using the [cbdocloader](#) tool. Example:  
`#/opt/couchbase/bin/cbdocloader -c localhost:8091 -u Administrator -p admin123 -b beer-sample -m 100 -d /opt/couchbase/samples/beer-sample.zip`

```
[root@tos bin]# /opt/couchbase/bin/cbdocloader -c localhost:8091 -u Administrator -p admin123 -b beer-sample -m 100 /opt/couchbase/samples/beer-sample.zip
Warning: Specifying the dataset without the -d/--dataset option is deprecated
Data loaded successfully
[root@tos bin]#
```

-m parameter will allocate 100 MB to the above bucket.

The above command will create beer-sample bucket and load the documents on it.

After that, you can verify from the web console:

name ▾	items	resident	ops/sec	RAM used/quota	disk used	<a href="#">Documents</a>	<a href="#">Statistics</a>
beer-sample	7,305	100%	0	27.4MB / 100MB	22.4MB	<a href="#">Documents</a>	<a href="#">Statistics</a>
couchmusic2	213,063	100%	0	268MB / 356MB	232MB	<a href="#">Documents</a>	<a href="#">Statistics</a>
default	1	100%	0	22.9MB / 256MB	4.08MB	<a href="#">Documents</a>	<a href="#">Statistics</a>
game-sample	586	100%	0	22.8MB / 100MB	5.05MB	<a href="#">Documents</a>	<a href="#">Statistics</a>

Execute the bucket list again:

```
#!/couchbase-cli bucket-list -c localhost -u Administrator -p admin123
```

```
[root@mymaster bin]# ./couchbase-cli bucket-list -c 127.0.0.1:8091 -u Administrator -p admin123
beer-sample
bucketType: membase
authType: sasl
saslPassword:
numReplicas: 1
ramQuota: 104857600
ramUsed: 55268848
default
bucketType: membase
authType: sasl
saslPassword:
numReplicas: 0
ramQuota: 104857600
ramUsed: 49205016
[root@mymaster bin]#
```

Flushes all data from the disk for a given bucket.

```
./couchbase-cli bucket-flush -c tos.hp.com:8091 -u Administrator -p admin123 --bucket=beer-sample
```

```
ramUsed: 40726120
[root@tos bin]# ./couchbase-cli bucket-flush -c tos.hp.com:8091 -u Administrator -p admin123 --bucket=beer-sample
Running this command will totally PURGE database data from disk. Do you really want to do it? (Yes/No)yes
ERROR: _ - Flush is disabled for the bucket
[root@tos bin]#
```

You will get an error if flush is not enable for that particular bucket.

Let us enable it before executing again. Click on Buckets → The bucket [beer-sample] --> Edit In the Advanced bucket settings Section.

Default    High

#### Auto-Compaction

Override the default auto-compaction settings?

#### Flush

Enable

Cancel

Save Changes

And click Save Changes.

Execute the command now

```
[root@tos bin]# ./couchbase-cli bucket-flush -c tos.hp.com:8091 -u Administrator -p admin123 --bucket=beer-sample
Running this command will totally PURGE database data from disk. Do you really want to do it? (Yes/No) yes
SUCCESS: Bucket flushed
[root@tos bin]#
```

You can verify the content as shown below:

name ▾	items	resident	ops/sec	RAM used/quota	disk used
beer-sample	0	100%	0	21.6MB / 100MB	9.97MB
travel-sample	31,592	100%	0	38.8MB / 100MB	137MB

Let us load it again before going ahead; we will be using it in the later tutorials:

```
#/opt/couchbase/bin/cbdocloader -c localhost:8091 -u Administrator -p admin123 -b beer-sample -m 100 -d /opt/couchbase/samples/beer-sample.zip
```

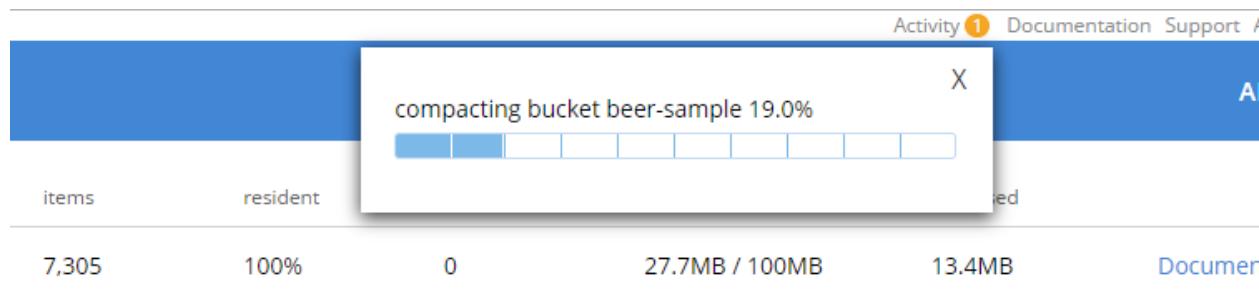
```
[root@tos bin]# /opt/couchbase/bin/cbdocloader -c localhost:8091 -u Administrator -p admin123 -b beer-sample -m 100 /opt/couchbase/samples/beer-sample.zip
Warning: Specifying the dataset without the -d/--dataset option is deprecated
Errors occurred during the index creation phase. See logs for details.
[root@tos bin]#
```

To compact a bucket for both data and view:

```
./couchbase-cli bucket-compact -c tos.hp.com:8091 -u Administrator -p admin123 --bucket=beer-sample
```

```
[root@mymaster bin]# ./couchbase-cli bucket-compact -c tos.master.com:8091 -u Ad
ministrator -p admin123 --bucket=beer-sample
SUCCESS: bucket-compact
[root@mymaster bin]#
```

You can verify the status on web console as shown below ( Click on the Activity button)



## cbstats

Retrieve lower level statistics with the cbstats utility, which provides deep insight into what occurs within a cluster.

```
/opt/couchbase/bin/cbstats -b beer-sample localhost:11210 all -u Administrator -p admin123 | grep \
curr_items:
```

When using the Couchbase data port 11210, it will give you operations per node per bucket:

```
[root@mymaster bin]# /opt/couchbase/bin/cbstats -b beer-sample localhost:11210 a
ll | grep \ curr_items:
 curr_items: 7303
[root@mymaster bin]#
```

The maximum amount of vbuckets that can exist in this bucket.

```
/opt/couchbase/bin/cbstats -b beer-sample localhost:11210 all | grep \ ep_max_vbuckets:
```

```
[root@mymaster bin]# /opt/couchbase/bin/cbstats -b beer-sample localhost:11210 a
ll | grep \ ep_max_vbuckets
 ep_max_vbuckets: 1024
[root@mymaster bin]#
```

Number of replica vBuckets.

```
/opt/couchbase/bin/cbstats -b beer-sample localhost:11210 all -u Administrator -p admin123 | grep \
ep_max_vbuckets:
```

```
[root@tos bin]# /opt/couchbase/bin/cbstats -b beer-sample localhost:11210 all -u Administrator -p admin123 | grep \ ep_max_vbuckets
ep_max_vbuckets: 1024
[root@tos bin]#
```

To delete the Test bucket:

Create a test\_bucket before performing the next command.

```
./couchbase-cli bucket-delete -c tos.hp.com:8091 -u Administrator -p admin123 \
--bucket=test_bucket
```

```
[root@tos bin]# ./couchbase-cli bucket-delete -c tos.hp.com:8091 -u Administrator -p admin123 \
> --bucket=test_bucket
SUCCESS: Bucket deleted
[root@tos bin]#
```

Determine a key from the bucket beer-sample using the following option. Get the vBucket of that doc ID.

Refer the following step:

21st\_amendment\_brewery\_cafe

The screenshot shows the Couchbase Web UI interface. On the left, there's a sidebar with navigation links: Dashboard, Servers, Buckets, XDCR, Security, Settings, Logs, Documents (which is selected), Query, and Search. The main area has a blue header with the title 'Documents'. Below the header, there are search and filter controls: 'Bucket' set to 'beer-sample', 'Limit' set to 10, 'Offset' set to 0, 'Document ID' input field, 'Where' query input field, and a 'Retrieve Docs' button. Underneath these controls, it says '10 Results for beer-sample, limit: 10, offset: 0'. The results list shows several documents, each with a row of icons (edit, preview, delete, copy) and a link to the document details. One document, '21st\_amendment\_brewery\_cafe', is circled in red. The document details show its ID, some metadata, and a long JSON object describing its properties.

```
./cbc-hash 21st_amendment_brewery_cafe -U couchbase://tos.hp.com/beer-sample -u Administrator -P admin123
```

```
e timeout (0x17)
[root@tos bin]# ./cbc-hash 21st_amendment_brewery_cafe -U couchbase://tos.hp.com/beer-sample -u Administrator -P admin123
21st_amendment_brewery_cafe: [vBucket=553, Index=0] Server: tos.hp.com:11210, CouchAPI: http://tos.hp.com:8092/beer-sample
Replica #0: Index=-1, Host=N/A
[root@tos bin]#
```

In this example, the key is in vBucket : 553

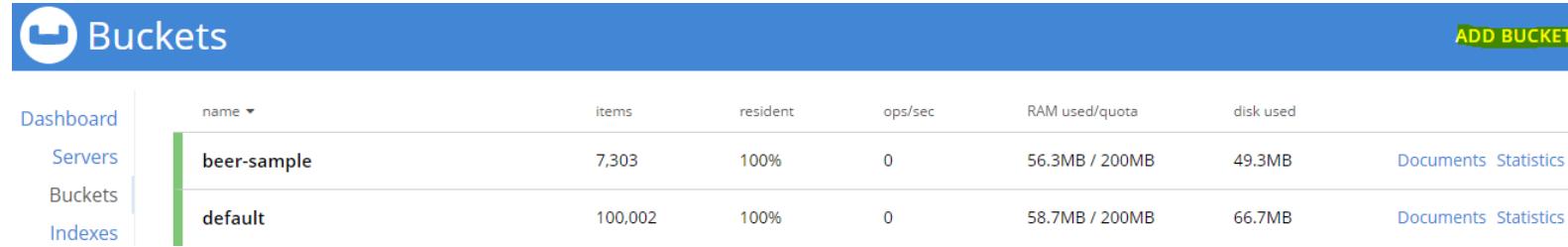
-----End of Lab-----

## 8. Bucket , Scope and Collections- 60 Minutes (D)

Logon the Web Console.

To Create New Bucket

Access the web console -> Buckets -> Add bucket.



name ▾	items	resident	ops/sec	RAM used/quota	disk used	Documents	Statistics
beer-sample	7,303	100%	0	56.3MB / 200MB	49.3MB	<a href="#">Documents</a>	<a href="#">Statistics</a>
default	100,002	100%	0	58.7MB / 200MB	66.7MB	<a href="#">Documents</a>	<a href="#">Statistics</a>

Enter the information as below:

Bucket Name: MyBucket

Bucket RAM Quota : 100 MB

Bucket Type : couchbase

### Add Data Bucket

X

Name  
MyBucket

Memory Quota in megabytes per server node  
100 MB

other buckets (600 MB)    this bucket (200 MB)    remaining (224 MB)

Bucket Type  
 Couchbase    Memcached    Ephemeral

▶ Advanced bucket settings

Cancel   Add Bucket

Click on Advanced Bucket settings, to configure setting of the bucket.  
You should be able to view the Bucket as below:

**▼ Advanced bucket settings****Replicas** Enable  Number of replica (backup) copies

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

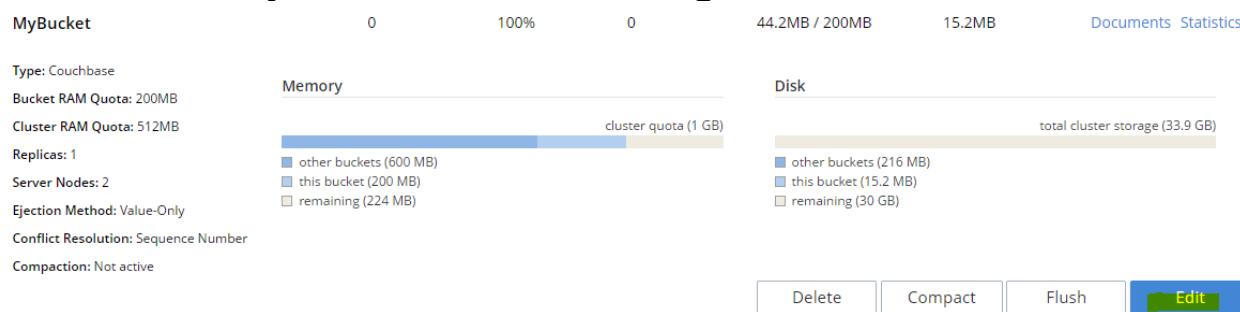
 Replicate view indexes**Bucket Max Time-To-Live ⓘ** Enable  seconds**Compression Mode ⓘ** Off  Passive  Active**Conflict Resolution ⓘ** Sequence number  Timestamp**Ejection Method ⓘ** Value-only  Full**Bucket Priority ⓘ** Default  High**Auto-Compaction ⓘ** Override the default auto-compaction settings?**Flush ⓘ** Enable

Click on Add Bucket.

You can find the new bucket in the bucket section. Click on Buckets menu.

	name ▾	items	resident	ops/sec	RAM used/quota	disk used	<a href="#">Documents</a>	<a href="#">Statistics</a>
Dashboard	beer-sample	7,303	100%	0	16.8MB / 100MB	34.3MB	<a href="#">Documents</a>	<a href="#">Statistics</a>
Servers	default	100,000	100%	0	32.4MB / 100MB	9.57MB	<a href="#">Documents</a>	<a href="#">Statistics</a>
Buckets	MyBucket	0	100%	0	20.5MB / 100MB	4.05MB	<a href="#">Documents</a>	<a href="#">Statistics</a>

Click on the MyBucket to edit the setting of the bucket.



Using this option increases the RAM from 100 to 102 MB.

Here, enable Flush so that a command can be issued to the bucket that can delete all documents in that bucket.  
Congrats, in having your own bucket!

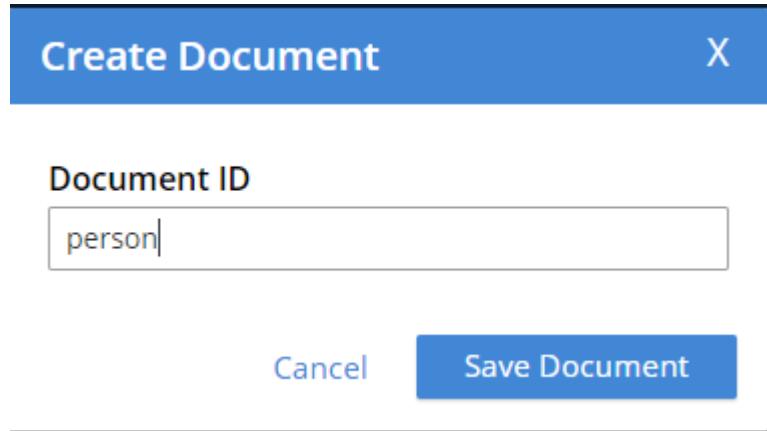
Let us add a few documents in our bucket, which you have just created.

Click on Buckets -> “MyBucket” -> Documents

	name ▾	items	resident	ops/sec	RAM used/quota	disk used	<a href="#">Documents</a>	<a href="#">Statistics</a>
beer-sample	beer-sample	7,303	100%	0	56.3MB / 200MB	49.3MB	<a href="#">Documents</a>	<a href="#">Statistics</a>
default	default	100,002	100%	0	58.7MB / 200MB	66.7MB	<a href="#">Documents</a>	<a href="#">Statistics</a>
MyBucket	MyBucket	0	100%	0	44.2MB / 200MB	18.9MB	<a href="#">Documents</a>	<a href="#">Statistics</a>
travel-sample	travel-sample	31,592	100%	0	119MB / 200MB	100MB	<a href="#">Documents</a>	<a href="#">Statistics</a>

>> Documents ->> Add Document

Enter Document ID : person



---

**Click Save Document.**

Let us modify the document as below:

```
{  
  "name": "Henry P",  
  "profession": "Free Lancer",  
  "city" : "Mumbai",  
  "pincode": 400063,  
  "skills" : ["Couchbase","Hadoop","Blockchain"]  
}
```

person

```
1 {
2   "name": "Henry P",
3   "profession": "Free Lancer",
4   "city" : "Mumbai",
5   "pincode": 400063,
6   "skills" : ["Couchbase", "Hadoop", "Blockchain"]
7 }
```

Click on save.

You can view the document as below:

Buckets → MyBucket → Documents → Enter “**person**” in text box(Document ID) and click “Retrieve Docs”

You are finding the document by its Id, person

Add one more document with your details.

Create the following documents in the “MyBucket”

Data Buckets -> MyBucket -> Documents

```

1 {
2   "type": "dept",
3   "id": 10,
4   "name": "Accounting",
5   "city": "New York"
6 }

```

The screenshot shows the Couchbase Admin UI with the navigation bar "MyBucket > Documents". A specific document named "emp\_20" is selected. The document content is displayed in a code editor-like interface:

```

1  {
2   "type": "emp",
3   "id": 7782,
4   "name": "Blake",
5   "job": "Clark",
6   "manager": 7839,
7   "salary": 2450,
8   "dept_id": "dept_10"
9 }

```

You should have the following documents as follows:

The screenshot shows the Couchbase Admin UI displaying three documents in the "MyBucket" collection:

ID	Content	Edit Document	Delete
Person	{ "name": "Henry P", "Profession": " Freelance Consultant/Trai..."}	Edit Document	Delete
dept_10	{ "type": "dept", "id": 10, "name": "Accounting", "city": "New..."}	Edit Document	Delete
emp_20	{ "type": "emp", "id": 7782, "name": "Blake", "job": "Clark", ...}	Edit Document	Delete

## Change Bucket Settings

Use the Mybucket which was created earlier to modify some of its settings. Let us increase the RAM size to 120 MB.

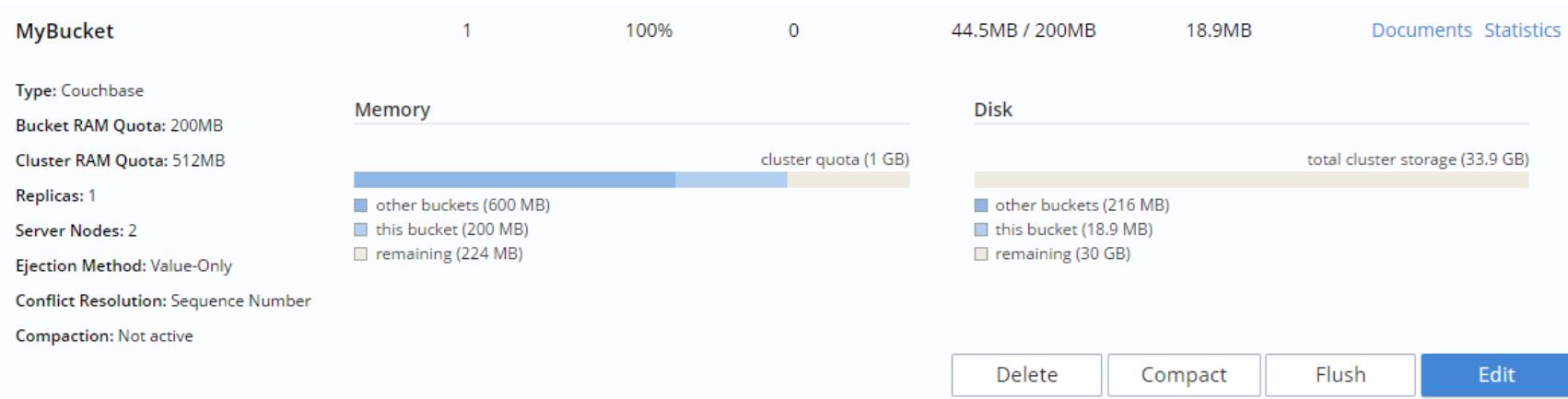
Add a couple of documents in the bucket you have created, Mybucket.

Full, Cluster and Bucket Administrators can edit bucket settings using the Change Bucket Settings option.

To edit the bucket settings:

Select >Buckets > MyBucket

Click on the **Edit** button.



The Configure Bucket dialog offers the same options as the [Create a New Bucket](#) dialog.

You can also change bucket settings with the CLI command [bucket-edit](#) or the REST command [rest-bucket-parameters](#).

**Edit Bucket Settings**

**Name**  
MyBucket

**Memory Quota** in megabytes per server node  
120 MB

other buckets (600 MB)    this bucket (240 MB)    remaining (184 MB)

**Bucket Type**  
 Couchbase     Memcached     Ephemeral

► Advanced bucket settings

[Cancel](#) [Save Changes](#)

Click save changes.

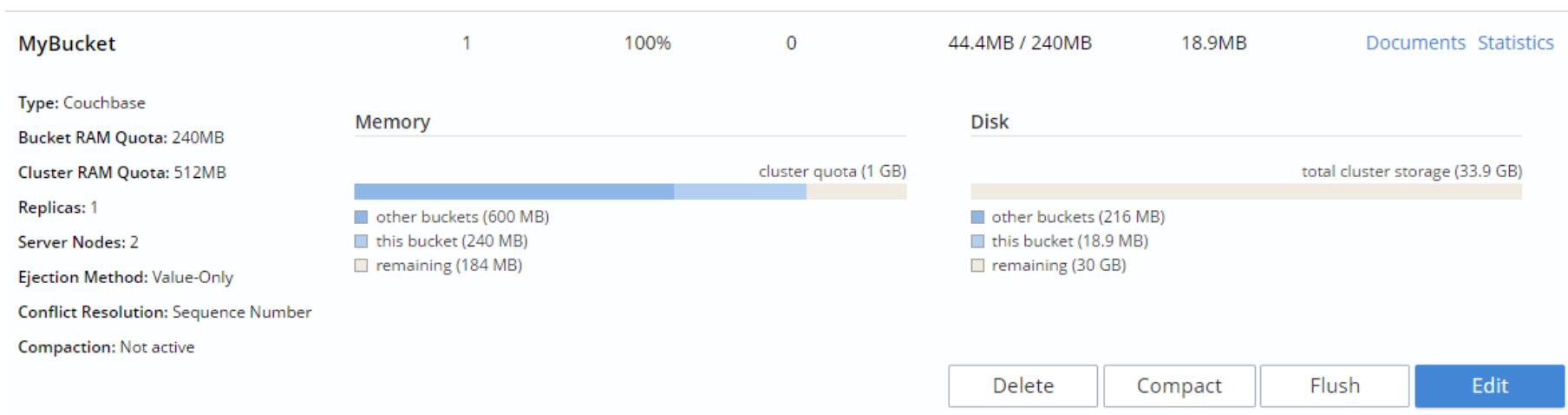
### Flush a Bucket

Full, Cluster, and Bucket Administrators can use the **Flush** button to start bucket flushing and delete every object in a bucket.

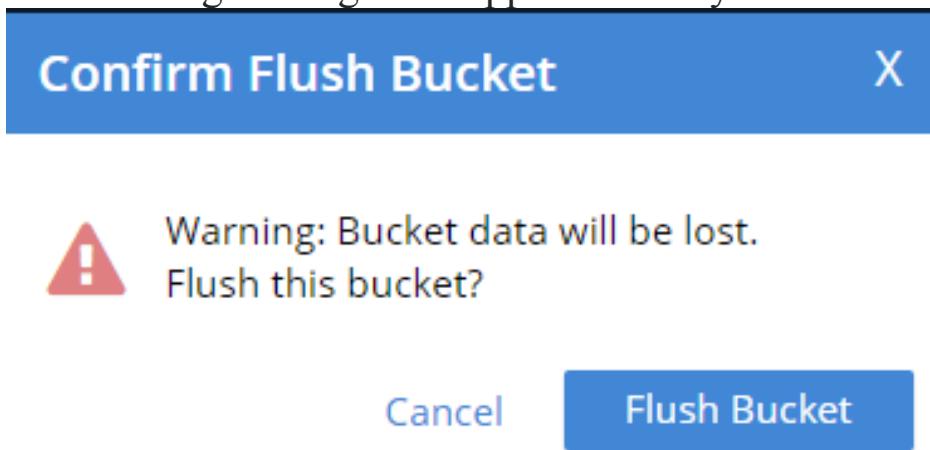
Flushing will work only if this option was configured either during the initial [bucket setup](#) or after the [bucket settings](#) have been changed.

Go to the Buckets and click the bucket name.

Click on the **Flush** button.



The warning message will appear before you flush the bucket.



Click the **Flush** button to start flushing. Confirm the no of records in that bucket

name ▾	items	resident	ops/sec
beer-sample	7,303	100%	0
default	100,002	100%	0
MyBucket	0	100%	0
travel-sample	31,592	100%	0

It should be 0.

## Delete a Bucket

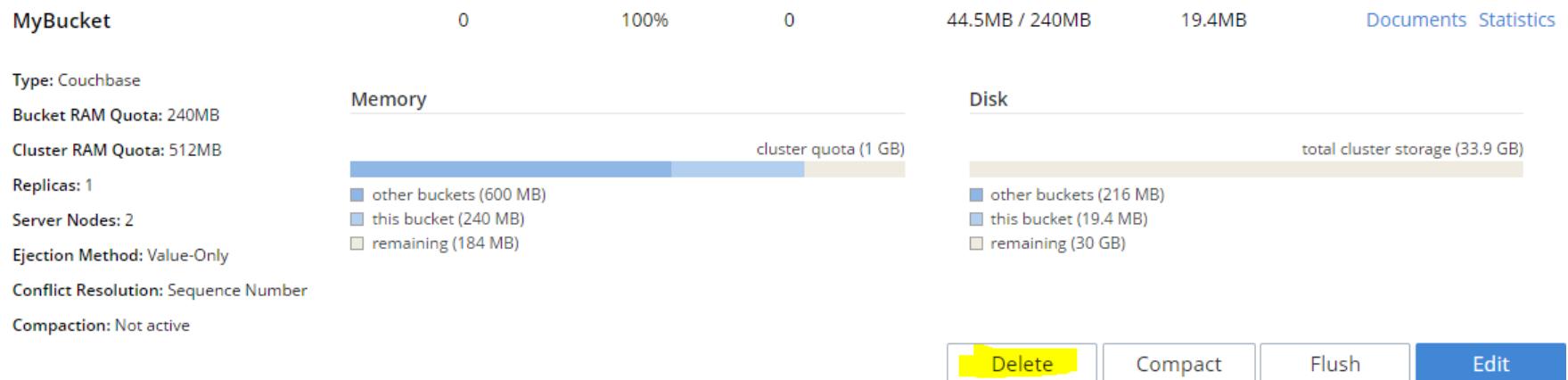
Full Administrators and Cluster Administrators can delete a bucket.

Deleting a bucket and then recreating it is sometimes faster than deleting all objects in an existing bucket.

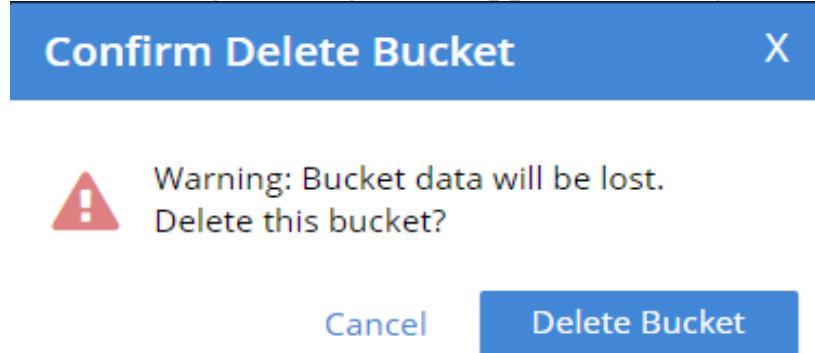
To delete a bucket using the Couchbase Web Console:

Go to the Data Buckets tab and click on the bucket name.

Click on the **Delete/Drop** button.



The warning message will appear before you remove the bucket.



Click on **Delete** and confirm the deletion.

Verify that MyBucket is not being displayed in the Bucket view.

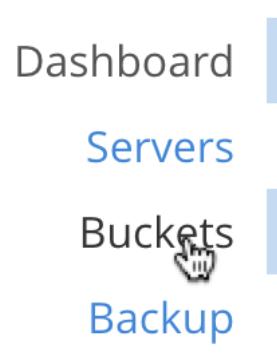
name ▾	items	resident	ops/sec	RAM used/quota	disk used	Documents	Statistics
beer-sample	7,303	100%	0	56.3MB / 200MB	49.3MB	Documents	Statistics
default	100,002	100%	0	58.8MB / 200MB	66.7MB	Documents	Statistics
travel-sample	31,592	100%	0	119MB / 200MB	100MB	Documents	Statistics



# Manage Scopes and Collections with the UI

The following sequence demonstrates how to create, examine, and delete scopes and collections, using the UI of Couchbase Web Console. Proceed as follows:

- 1 Access Couchbase Web Console, and left-click on the **Buckets** tab, in the vertical, left-hand navigation bar:



- 2 This brings up the **Buckets** screen.

- 3 Add a new bucket, named **testBucket**. Left-click on the **ADD BUCKET** tab, at the upper right:



This brings up the **Add Data Bucket** dialog. Use this to create **testBucket** as a Couchbase bucket of 256 MB, as follows:

## Add Data Bucket

**Name**  
testBucket

**Memory Quota** in megabytes per server node  
256 MiB

other buckets (600MiB)    this bucket (768MiB)    available (168MiB)

**Bucket Type**  
 Couchbase    Memcached    Ephemeral

► Advanced bucket settings

[Cancel](#) [Add Bucket](#)

Left-click on the **Add Bucket** button, to create. The **Buckets** screen now displays the newly created bucket:

filter buckets...		name ▲	items	resident	ops/sec	RAM used/quota	disk used	Documents	Scopes & Collections
testBucket			0	100%	0	0B / 512MiB	0B		

4 Examine the new bucket's *default collection*. At the right-hand side of the row that displays the new bucket, two options appear: **Documents** and **Scopes & Collections**. Left-click on the **Scopes & Collections** option:



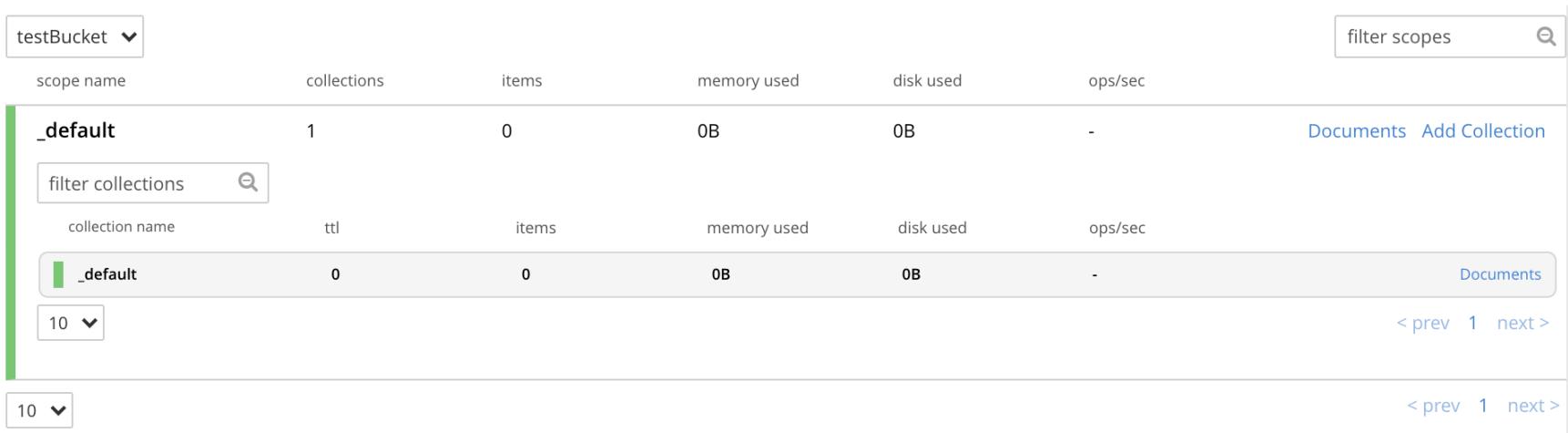
This brings up the **Scopes & Collections** screen, which appears as follows:

scope name	collections	items	memory used	disk used	ops/sec	
_default	1	0	0B	0B	-	<a href="#">Documents</a> <a href="#">Add Collection</a>

filter scopes

< prev 1 next >

The screen features a single row, which confirms the existence of the **\_default** scope. As its name indicates, this scope is created by default for each new bucket. Collections that are created without any administrator-defined scope specified as their destination are duly saved in the **\_default** scope. To examine the contents of the **\_default** scope, left-click on the row. The row expands as follows:



The screenshot shows the 'testBucket' scope details. It includes a table with columns: scope name, collections, items, memory used, disk used, and ops/sec. A sub-table for the '\_default' collection shows collection name, ttl, items, memory used, disk used, and ops/sec. Navigation controls like 'filter scopes', 'Documents', 'Add Collection', and pagination (10, < prev, 1, next >) are also visible.

scope name	collections	items	memory used	disk used	ops/sec	
<b>_default</b>	1	0	0B	0B	-	<a href="#">Documents</a> <a href="#">Add Collection</a>
collection name	ttl	items	memory used	disk used	ops/sec	
<b>_default</b>	0	0	0B	0B	-	<a href="#">Documents</a>
10						< prev 1 next >
10						< prev 1 next >

This indicates that the **\_default** scope contains a single collection, which is the **\_default** collection. As its name indicates, this collection is created by default, within the **\_default** scope, for each new bucket. Documents that are created without any administrator-defined collection specified as their destination are duly saved in the **\_default** collection.

5 Add a scope to the bucket. Left-click on the **ADD SCOPE** tab, at the upper right of the screen:



6 This brings up the **Add Scope** dialog, which appears as follows:

## Add Scope

X

**Bucket Name**

**New Scope Name**

---

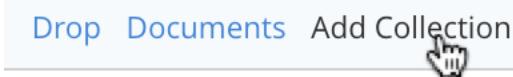
[Cancel](#) [Save](#)

7 Enter the name of a new scope into the **New Scope Name** field: for example, **MyScope**. Then, left-click on the **Save** button. The **Scopes & Collections** screen now appears as follows:

scope name	collections	items	memory used	disk used
MyScope	0	-	-	-
_default	1	0	0B	0B
<input type="text" value="filter collections"/>				
collection name	ttl	items	memory used	disk used
_default	0	0	0B	0B
10				
10				

8 The new scope, **MyScope**, is now displayed.

9 Add a new collection, within the new scope. At the extreme right of the row for **MyScope** are two options: **Drop** and **Add Collection**. Left-click on **Add Collection**:



10 This brings up a dialog entitled **Add Collection into MyScope scope**:

Add Collection into MyScope scope X

**Name**

**Collection Max Time-To-Live (i)**

[Cancel](#) [Save](#)

- 11 Enter the name of a new collection into the **Name** field: for example, **MyCollection**. The **Collection Max Time-To-Live** field can, for this example, be left at its default value of 0 — which indicates that no *expiration* value is assigned to the collection and the documents it contains. (For information, see [Expiration](#)). Left-click on the **Save** button. Next, left-click on the row for **MyScope**.

The row expands, as follows:

scope name	collections	items	memory used	disk used	ops/sec	
MyScope	1	-	-	-	-	<a href="#">Drop</a> <a href="#">Documents</a> <a href="#">Add Collection</a>
	<input style="width: 150px; height: 25px; border: 1px solid #ccc; border-radius: 5px; padding: 2px 10px; margin-right: 10px;" type="text" value="filter collections"/>					
	collection name	ttl	items	memory used	disk used	ops/sec
	<input style="width: 150px; height: 25px; border: 1px solid #ccc; border-radius: 5px; padding: 2px 10px; margin-right: 10px;" type="text" value="MyCollection"/>	0	-	-	-	<a href="#">Drop</a> <a href="#">Documents</a>
	<input style="width: 50px; height: 25px; border: 1px solid #ccc; border-radius: 5px; padding: 2px 10px; margin-right: 10px;" type="button" value="10"/>					< prev 1 next >

- 12 The new collection, **MyCollection**, is thus shown to have been created within the scope **MyScope**.

To add document in the collection.

Click on the Document of the collection → Add document.

The screenshot shows the Couchbase Workbench interface. At the top, there's a blue header bar with the text "tos.couchbase - Enterprise Edition 7.0.3 build 7031" and "Workbench Import". Below the header, there's a search bar with dropdown menus for "Keyspace" (set to "testBucket"), "Scope" (set to "MyScope"), and "Collection" (set to "MyCollection"). To the right of the search bar are buttons for "Limit" (set to 10), "Offset" (set to 0), "Document ID" (with a placeholder "optional..."), "show range", "N1QL WHERE", and "Retrieve Docs". A red circle highlights the "ADD DOCUMENT" button in the top right corner. Below the search bar, a message says "1 Results for testBucket.MyScope.MyCollection, limit: 10, offset: 0". There's a toggle switch for "enable field editing" which is turned off. At the bottom, a table displays a single document with the ID "cb". The table has columns for "id" and "cb". The "cb" row contains the JSON object {"learn": "couchbase"}. Below the table are icons for edit, delete, and other actions. A red line highlights the "cb" ID in the table.

ID: cb

```
{  
  "learn": "couchbase"  
}
```

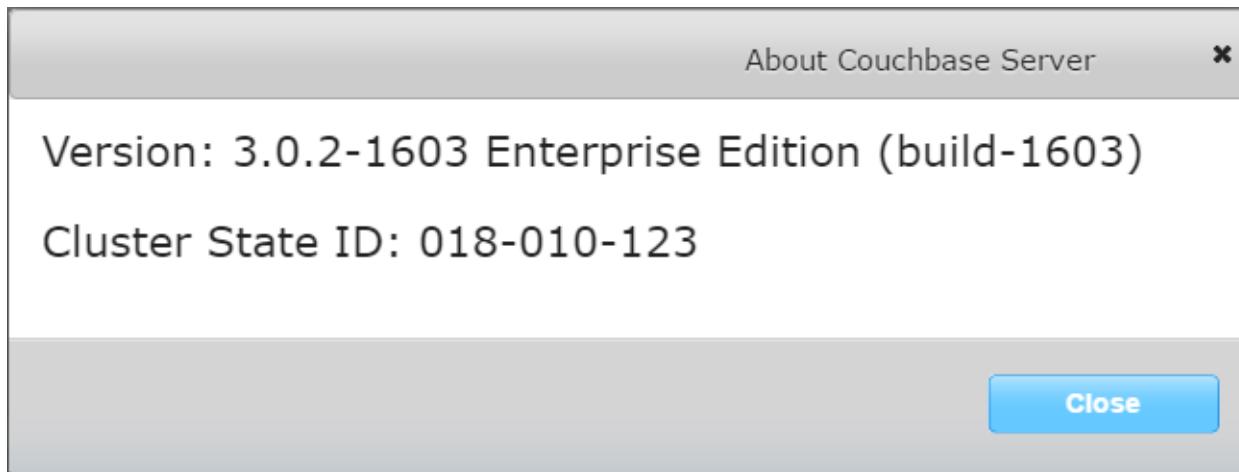
-----End of Lab-----

## 9. Upgrade - TBD

Install Couchbase server 3.0



Accept all default values and verify the version by clicking on About which is on the right top corner of the web console



Create a bucket : Henry

Data Buckets

Couchbase Buckets							Create New Data Bucket	
Bucket Name	Nodes	Item Count	Ops/sec	Disk Fetches/sec	RAM/Quota Usage	Data/Disk Usage	Documents	Views
▶ default	● 1	0	0	0	31.3MB / 100MB	2.36MB / 2.27MB	Documents	Views
▶ henry	● 1	0	0	0	30.6MB / 100MB	0B / 34B	Documents	Views

And create two documents:

ID : skills

{

```
"name" : " Couchbase Administration"
}
```

ID : location

```
{
  "city" : " Mumbai"
}
```

Couchbase Buckets

Bucket Name	Nodes	Item Count	Ops/sec	Disk Fetches/sec	RAM/Quota Usage	Data/Disk Usage	Documents	Views
▶ default	1	0	0	0	30.8MB / 100MB	4.04MB / 4.05MB	Documents	Views
▶ henry	1	2	0	0	31.5MB / 100MB	4MB / 3.91MB	Documents	Views

henry > Documents

Current page: 1 2 3 4 5 6

Documents Filter ▾

ID	Content	Edit Document	Delete
location	{ "city": " Mumbai" }	Edit Document	Delete
skills	{ "name": " Couchbase Administration" }	Edit Document	Delete

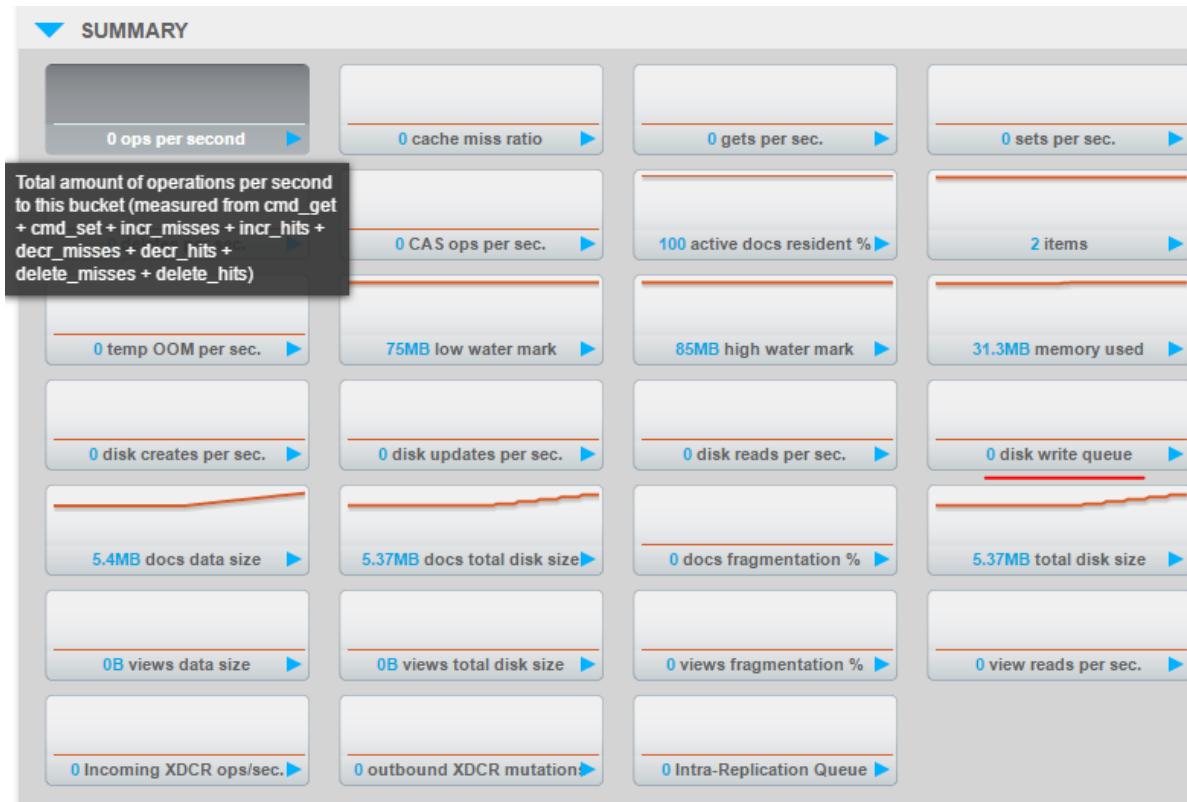
To perform an offline upgrade:

1. Disable auto-failover for all nodes in the cluster. Use the Couchbase Web Console under **Settings > Auto-Failover**.

If you do not disable auto-failover, the first node that you shut down will be failed over automatically by the cluster.

The screenshot shows the Couchbase Web Console interface. At the top, there is a navigation bar with tabs: Cluster Overview, Server Nodes, Data Buckets, Views, XDCR, Log, and Settings. The Settings tab is currently selected. Below the navigation bar, there is a sub-navigation bar with tabs: Cluster, Update Notifications, Auto-Failover, Alerts, Auto-Compaction, Sample Buckets, and Account Management. The Auto-Failover tab is selected. The main content area is titled "Auto-Failover". It contains two configuration options: "Enable auto-failover" (with a checked checkbox) and "Timeout: 120" (with a "What's this?" link). At the bottom right of the content area is a blue "Save" button.

2. You can use the Couchbase Web Console to monitor cluster activity. The cluster must finish writing all information to disk. When you restart your cluster, all of your data can be brought back into the caching layer from disk.  
You can monitor the Disk Write Queue statistics for each bucket in your cluster. When the queue reaches zero, no more data remains to be written to disk.
3. Open the Couchbase Web Console at a node in your cluster.
4. Click **Data Buckets** | *henry*. In the Summary section, check that Disk write queue is equal to 0. If you have more than one data bucket in your cluster, repeat this check on each bucket.

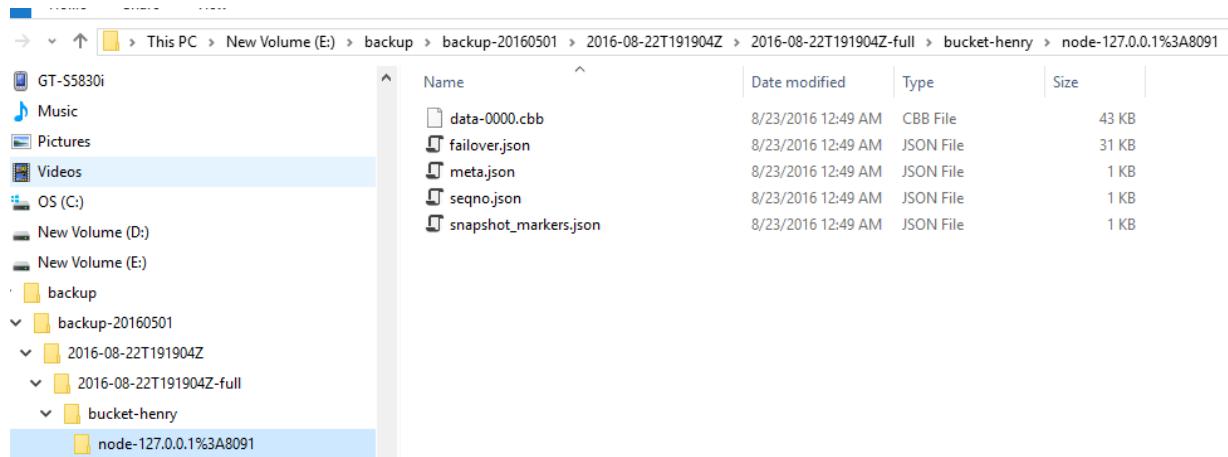


## 5. Create a backup of your cluster data using the [cbackup tool](#).

```
#cbackup http://localhost:8091 E:/backup/backup-20160501 -u Administrator -p admin123 -b henry
```

```
c:\Program Files\Couchbase>cd server
c:\Program Files\Couchbase\Server>cd bin
c:\Program Files\Couchbase\Server\bin>cbackup http://localhost:8091 E:/backup/backup-20160501 -u Administrator -p admin123 -b henry
[########################################] 100.0% (2/estimated 2 msgs)
bucket: henry, msgs transferred...
byte :      total |      last |    per sec
done
```

Backup folder is as shown below, We will discuss about back up in details later.



## 6. Stop the Couchbase Server service on each cluster node.

	CoreMessaging	Manages co...	Running	Automatic	Local Service
	CouchbaseServer	Couchbase ...		Automatic	Local Syste...
	Credential Manager	Provides se...	Running	Manual	Local Syste...

## 7. After you shut down the services, perform a standard node upgrade to the new version of Couchbase Server as explained in [Performing the Single Node Upgrade](#).

Back up the server-specific configuration files.

Name	Date modified	Type	Size
config	8/23/2016 12:37 AM	File folder	
data	8/23/2016 12:37 AM	File folder	
logs	8/23/2016 12:51 AM	File folder	
stats	8/23/2016 12:51 AM	File folder	
couchbase-server.node	8/23/2016 12:34 AM	NODE File	1 KB
initargs	8/23/2016 12:34 AM	File	3 KB
ip	12/4/2014 3:15 PM	File	0 KB
ip_start	12/4/2014 3:15 PM	File	0 KB
isasl.pw	8/23/2016 12:37 AM	PW File	1 KB
ns_log	8/23/2016 12:51 AM	File	2 KB
remote_clusters_cache_v3	8/23/2016 12:50 AM	File	1 KB

Copy it to another folder location:

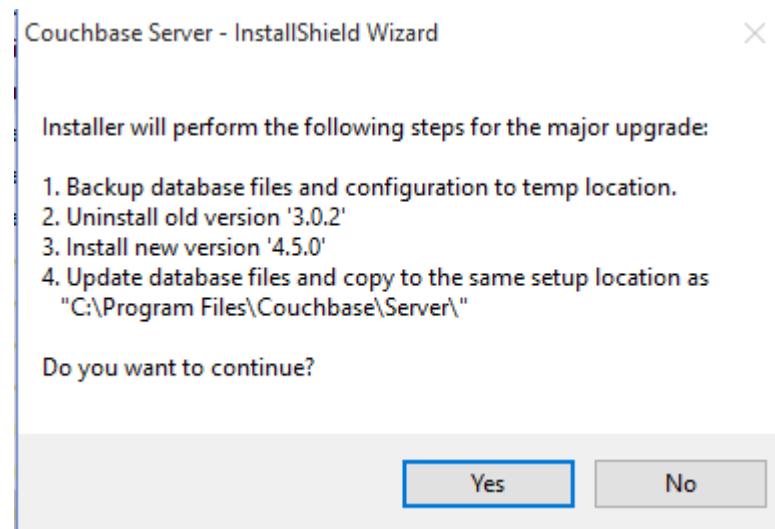
Name	Date modified	Type	Size
backup-20160501	8/23/2016 12:49 AM	File folder	
config	8/23/2016 12:53 AM	File folder	

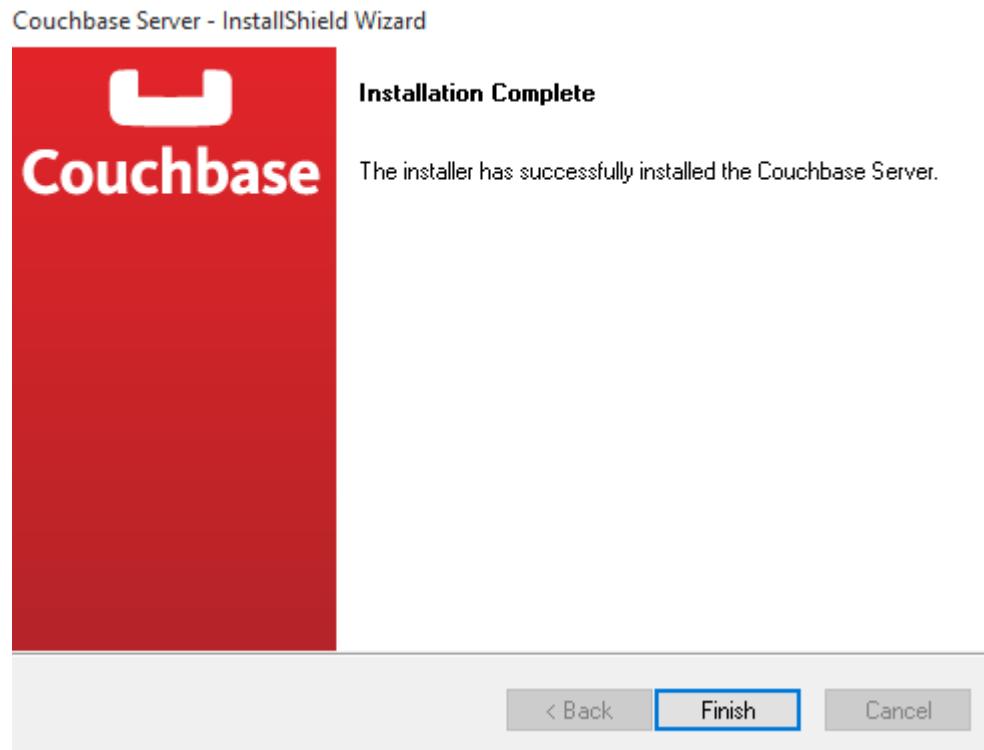
Install the couchbase 4.5 , by double click it

File	Date	Type	Size
couchbase-server-enterprise_4.5.0-windows_amd64.exe	8/19/2016 9:45 PM	Application	20

Accept all default value,

You will be prompted with the following option, accept ok to proceed.





Click Finish

Couchbase Server starts automatically on each node after you perform the node upgrade.

8. You can monitor the warmup process status after starting the cluster with the [cbstats tool](#) to determine when the cluster can begin servicing application requests.

```
C:\Program Files\Couchbase\Server\bin>cbstats -b henry localhost:11210 all
accepting_conns:          1
auth_cmds:                85
auth_errors:              0
```

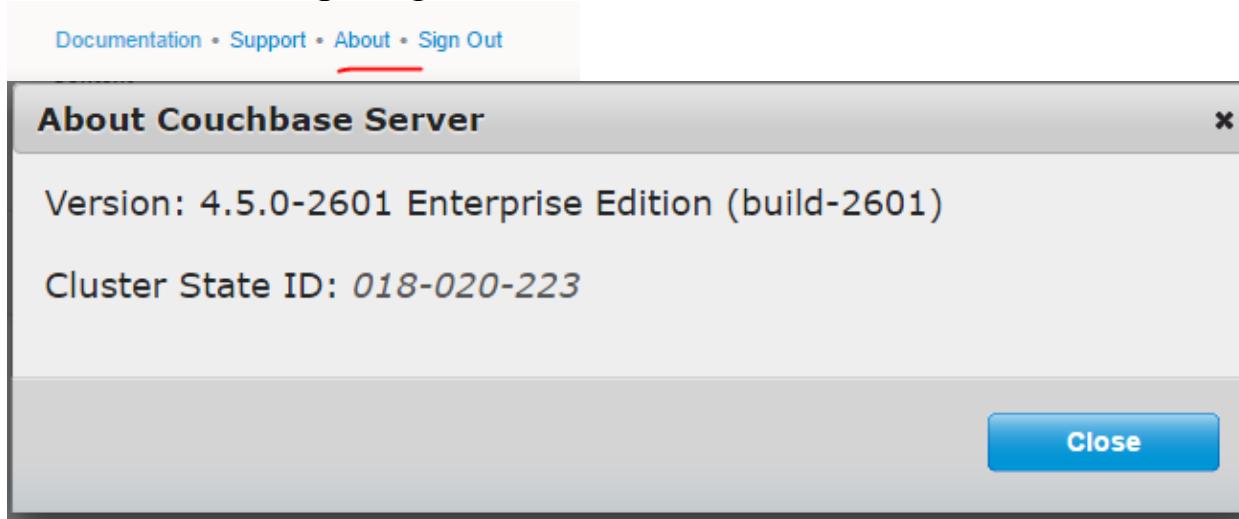
```
bytes_read:          10012
bytes_subdoc_lookup_extracted: 0
bytes_subdoc_lookup_total:    0
bytes_subdoc_mutation_inserted: 0
bytes_subdoc_mutation_total:   0
bytes_written:        6750
cas_badval:           0
cas_hits:             0
cas_misses:           0
cmd_flush:            0
cmd_get:              0
cmd_set:              0
cmd_subdoc_lookup:    0
cmd_subdoc_mutation:  0
cmd_total_gets:       0
cmd_total_ops:        0
cmd_total_sets:       0
conn_yields:          0
connection_structures: 11
curr_connections:     11
curr_conns_on_port_11207: 2
curr_conns_on_port_11209: 6
curr_conns_on_port_11210: 3
daemon_connections:   6
decr_hits:            0
decr_misses:          0
```

```
delete_hits:          0
delete_misses:        0
get_hits:             0
get_misses:           0
incr_hits:            0
incr_misses:          0
iovused_high_watermark: 2
libevent:              2.1.4-alpha-dev
listen_disabled_num:   0
max_conns_on_port_11207: 30000
max_conns_on_port_11209: 5000
max_conns_on_port_11210: 30000
memcached_version:     3333fbc2a5f27cd03d95e68b19599c38f657b14b
msgused_high_watermark: 1
pid:                  7232
pointer_size:          64
rbufs_allocated:        6
rbufs_existing:         86
rbufs_loaned:           350
rejected_conns:         0
stat_reset:             Tue Aug 23 01:00:57 2016
threads:                4
time:                  1471894277
total_connections:       91
uptime:                 20
version:                4.5.0-2601
```

```
wbufs_allocated:      6  
wbufs_loaned:       418
```

After the cluster finishes warmup, your applications can use the upgraded cluster.

You can verify the version after logon.  
click on About , right top corner.



Let us verify whether the henry bucket is intact.

Couchbase Buckets

Bucket Name	Data Nodes	Item Count	Ops/sec	Disk Fetches/sec	RAM/Quota Usage	Data/Disk Usage
▶ default	1	0	0	0	35.8MB / 100MB	8.04MB / 8.06MB
▶ henry	1	2	0	0	35.8MB / 100MB	8.05MB / 8.07MB

Create New Data Bucket

Click on the Documents --> henry bucket

Settings henry > Documents

Current page: 1 5

Documents Filter ▾

ID	Content	Actions
location	{"city": "Mumbai"}	Edit Document Delete
skills	{"name": "Couchbase Administration"}	Edit Document Delete

-----End Lab-----

## 10. Cluster Operations - 120 Minutes (D)

In this lab, we will understand how to add node to an existing cluster and perform balancing of documents. We will also perform some failover use cases.

Scenarios to be performed in this lab:

- Add Node to a cluster ( Use when you want to increase the cluster capacity)
- Hard Node Failure
  - Node Down and will be removed.
  - Shutdown Master Node
  - Failover master Node
  - Perform Rebalance.
- Failover of Node (Use when you will bring the node later after system maintenance)
  - Failover Slave Node.
  - Don't Perform rebalance
  - Perform Full recovery
  - Rebalance
- Removing Node (use when you want to reduce the cluster capacity)
  - Remove Slave Node
  - Rebalance

Let us set up a test two-node Couchbase environment.

### Using Docker or VM (Substitute Docker container instead of VM)

Determine the hostname and IPs of both the servers by logon to the servers. [hostname, ifconfig commands to determine the hostname and the ip address]

Master

IP Address	10.10.20.28
Hostname	tos.master.com / tos.hp.com / couchbase0.master.com

Now we need to set up the second node, which will be called as slave node.

We will change the hostname of the second node as slave, tos.slave.com

Open the following two files as shown below and update the hostname.

IP Address	10.10.20.29
Hostname	tos.slave.com / tos.slavea.com /couchbase1.master.com

[Hostname are bind at the time of installation]

Note: Refer the last page (Annexure) to un-installed the couchbase server if required.

Repeat the earlier steps to install Couchbase on the slave.

You have successfully installed Couchbase Server.  
Please browse to <http://tos.slave.com:8091/> to configure your server.  
Please refer to <http://couchbase.com> for additional resources.

Please note that you have to update your firewall configuration to allow connections to the following ports:  
1369, 8091 to 8094, 9100 to 9105, 9998, 9999, 11209 to 11211,  
11214, 11215, 18091 to 18093, and from 21100 to 21299.

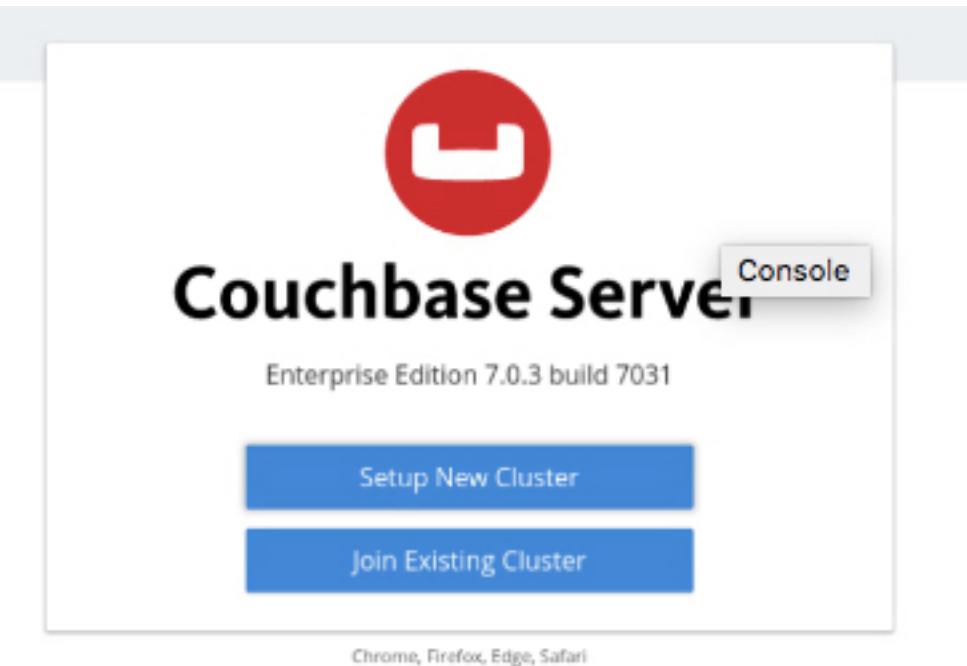
By using this software you agree to the End User License Agreement.  
See `/opt/couchbase/LICENSE.txt`.

## Add a Node during Initial Installation

**<http://tos.slave.com:8091/ui/index.html> [Ensure that you have entry of the hostname in your windows client]**

When setting up a new Couchbase Server installation, you have the option to join the new node to an existing cluster or start a new cluster.

During the first step, select the **Setup new cluster** button, as shown in the figure below:



You are prompted to add the following information:

**IP Address**

The IP address or hostname of an existing node within the cluster you want to join.

**Username**

The username of the administrator of the target cluster.

**Password**

The password of the administrator of the target cluster.

Note: You need to have entry on both the node so that each of the node are able to connect from each other using hostname.

In our case we will create a separate cluster and join it later on the earlier cluster. Select Setup New Cluster. Modify the following setting for the slave configuration:

Cluster Name	Slave Cluster
<b>Create Admin Username</b>	Administrator
<b>Create Password</b>	life213
Host Name / IP Address	tos.slave.com
<b>Couchbase Memory Quotas</b>	Data Service – 512 MB
Enable only Data services i.e unselect all other service	

Rest of the setting can remain the same. Accept all default and click Finish.

Verify the Couchbase server's administration console in your browser for both the servers.

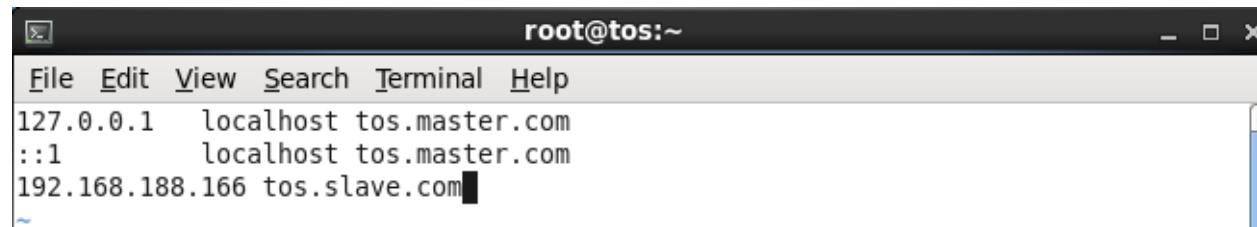
<http://tos.slave.com:8091/ui/index.html#/overview>

This time we will be joining to an existing cluster.

If you are unable to access the admin console, ensure to stop firewall on both the servers[note: service iptables stop]

Verify that couchbase server are up [service couchbase-server status or systemctl status couchbase-server]

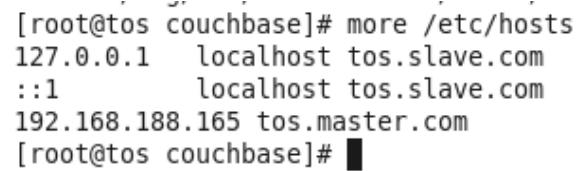
Ensure to include the slave details in the etc host of the master as shown below:



A screenshot of a terminal window titled "root@tos:~". The window has a menu bar with File, Edit, View, Search, Terminal, and Help. The main area displays the contents of the /etc/hosts file:

```
127.0.0.1 localhost tos.master.com
::1 localhost tos.master.com
192.168.188.166 tos.slave.com
```

And vice versa: On the hosts of Master.



A screenshot of a terminal window titled "[root@tos couchbase]# more /etc/hosts". The window shows the following content:

```
127.0.0.1 localhost tos.slave.com
::1 localhost tos.slave.com
192.168.188.165 tos.master.com
[root@tos couchbase]#
```

## Add Nodes via UI to join a cluster

Ensure that you have loaded beer-sample bucket in the master node before going ahead.

Perform the following activities on the master web UI

Hints : Add slave to the master.

<http://tos.master.com:8091/ui/index.html#!/overview>

To add a new node to an existing cluster after installation, click on Servers → **Add Server** button on the Couchbase Web Console.

name	group	services	CPU	RAM	swap	disk used	items
10.10.20.28	Group 1	data full text index query	17.5%	54%	0%	80.5MB	131 K/0

Using the same method, you can add a server that was previously part of this or another cluster. During this process, Couchbase Server must be running.

<b>Hostname/IP Address</b>	tos.slave.com
Username	Administrator / life213
Dataservice	Only this service. Unselect all others services.

Fill in the requested information:

### **Server IP Address**

The IP address or FQDN (Fully Qualified Domain Name) of the server that you want to add. It is preferred that you use FQDNs, especially in a cloud hosting environment.

### **Username**

The username of the administrator account of the node to be added.

### **Password**

The password of the administrator account of the node to be added.

### **Services**

By checking all appropriate check boxes (Data, Query, and Index), you can define what kind of [services](#) will be provided on the server node that's been added. You can add one, two, or all three services.

In our case add only Data Node.

**Add Server Node**

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

**Hostname/IP Address**  
tos.slavea.com

**Username** an existing username with admin access to this server  
Administrator

**Password** an existing password with admin access to this server  
.....

**Services** ⓘ  
 Data

Click on Add Server.

Or Using CLI :[

```
couchbase-cli server-add -c couchbase0.master.com:8091 \
--username Administrator \
--password admin123 \
--server-add http://couchbase1.master.com:8091 \
--server-add-username Administrator \
--server-add-password life213 \
--services data
```

]

## Using REST API

```
curl -u Administrator:admin123 -v -X POST \
couchbase0.master.com:8091/controller/addNode \
-d 'hostname=couchbase1.master.com&user=Administrator&password=life213&services=kv'
```

name	group	services	CPU	RAM	swap	disk used	items	Rebalance
10.10.20.28	Group 1	data full text index query	25.9%	54.6%	0%	80.5MB	131 K/0	<a href="#">Statistics</a>
tos.slave.com	Group 1	data	1.01%	24.3%	0%	---	0/0	<a href="#">Statistics</a>

New node | Not taking traffic | ADD pending rebalance

Cancel Add

Accept the warning and proceed.

Server should now be associated to your Couchbase cluster. Dashboard view

The screenshot shows the Tos.Couchbase dashboard with the following information:

- Services Summary:**
  - Data Service: 2 nodes (yellow)
  - Index Service: 1 node (green)
  - Search Service: 1 node (green)
  - Query Service: 1 node (green)
  - XDCR: 0 remote clusters, 0 replications
- Data Service Memory:** total quota (400 MB)
  - in use (176 MB)
  - unused quota (223 MB)
  - unallocated (824 MB)
- Data Service Disk:** usable free space (30.2 GB)
  - in use by couchbase (164 MB)
  - other data (3.4 GB)
  - free (30.4 GB)

## Servers view

The screenshot shows the Tos.Couchbase Servers view with the following details:

name	group	services	CPU	RAM	swap	disk used	items	Rebalance
10.10.20.28	Group 1	data full text index query	19.4%	52.5%	0%	80.5MB	131 K/0	<a href="#">Statistic:</a>
tos.slave.com	Group 1	data	1.01%	24.8%	0%	---	0/0	<a href="#">Statistic:</a>

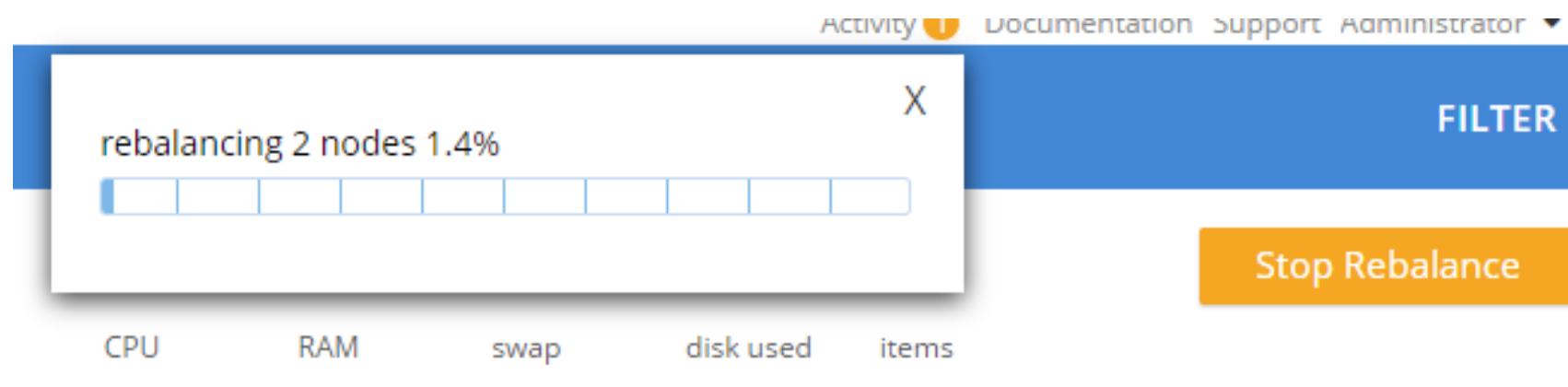
New node | Not taking traffic | ADD pending rebalance [Cancel Add](#)

Uptime: 27 minutes, 52 seconds  
 OS: x86\_64-unknown-linux-gnu  
 Version: Enterprise Edition 5.0.1 build 5003  
 Data Service RAM Quota: 612 MB  
 Data Storage Path: /opt/couchbase/var/lib/couchbase/data  
 Index Storage Path: /opt/couchbase/var/lib/couchbase/data

**Memory**  
 used (0 B) remaining (782 MB)

**Disk Storage**  
 used (0 B) remaining (15.4 GB)

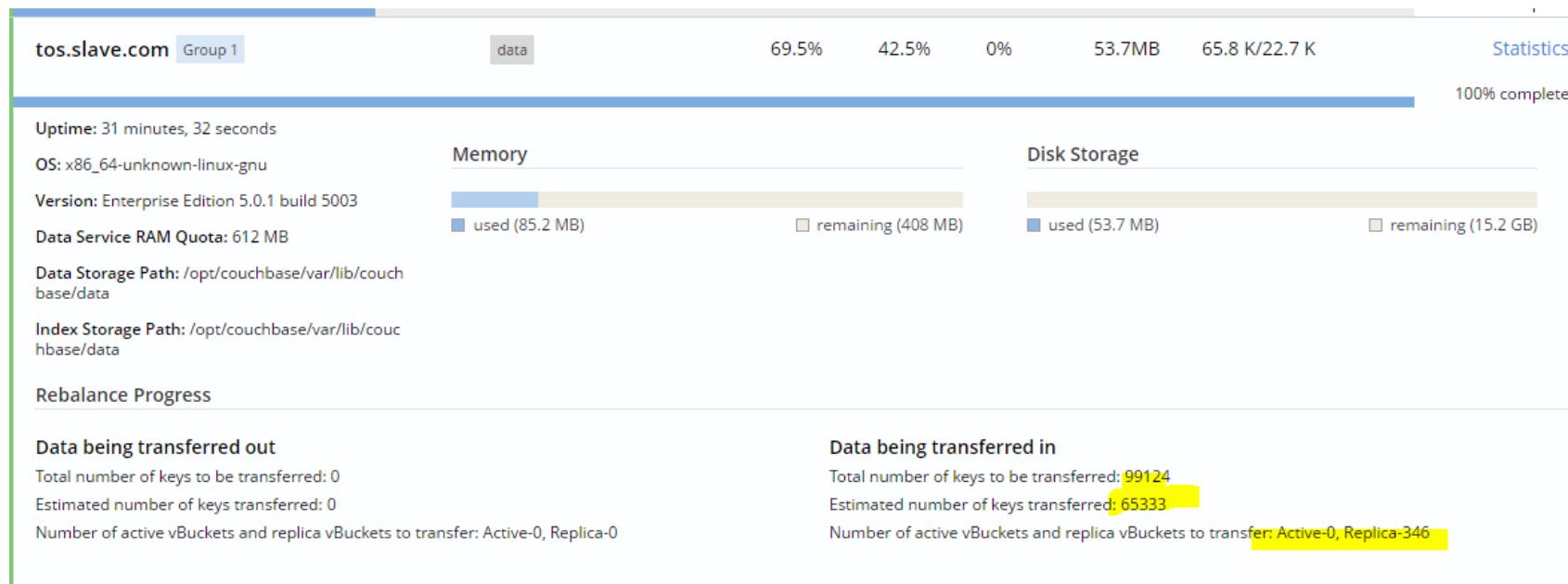
In order to actually use the new node with your cluster, the cluster needs to be rebalanced. Click “Server” from the left navigation and then click the “Pending Rebalance” tab. Then click “Rebalance” to the right. It will display the progress as shown below.



Wait for the nodes to rebalance before proceeding.

A screenshot of the Couchbase Server dashboard under the "Servers" tab. The sidebar on the left lists various cluster management options: Dashboard, Servers (selected), Buckets, Indexes, Search, Query, and XDCR. The main table displays two server nodes: "10.10.20.28" and "tos.slave.com". Both nodes are listed under "Group 1". The "10.10.20.28" row includes service status indicators for "data", "full text", "index", and "query". Resource usage metrics are shown for CPU (93.3%), RAM (59%), swap (0%), and disk used (85.7MB). A yellow progress bar indicates the rebalance progress at 18.2%, with a tooltip stating "3.7% complete". A "Statistics" link is available for this node. The "tos.slave.com" row shows similar metrics: CPU (77.5%), RAM (38%), swap (0%), and disk used (27.9MB). Its rebalance progress is at 15.8 K/8813, with a tooltip stating "32.7% complete". A "Statistics" link is also present here. A small progress dialog box is overlaid on the top right of the dashboard, showing "rebalancing 2 nodes 18.2%" with a matching progress bar.

You can click on the any of the server node to view the process: Then Data services.



When rebalancing is complete your nodes should look similar to this:

								<a href="#">Failover Multiple Nodes</a>	<a href="#">Rebalance</a>
name ▾	group	services			CPU	RAM	swap	disk used	items
192.168.139.129	Group 1	analytics	data	eventing	8.69%	69.8%	0%	54.9MB	53.7 K...
tos.slave.com	Group 1	index	query	search	4.68%	52.9%	2.23%	43.3MB	53.5 K...

Now it's time to fail some nodes.

## Failing over a Node

Failover is the process in which a node of a Couchbase cluster is removed quickly as opposed to a regular removal and rebalancing.

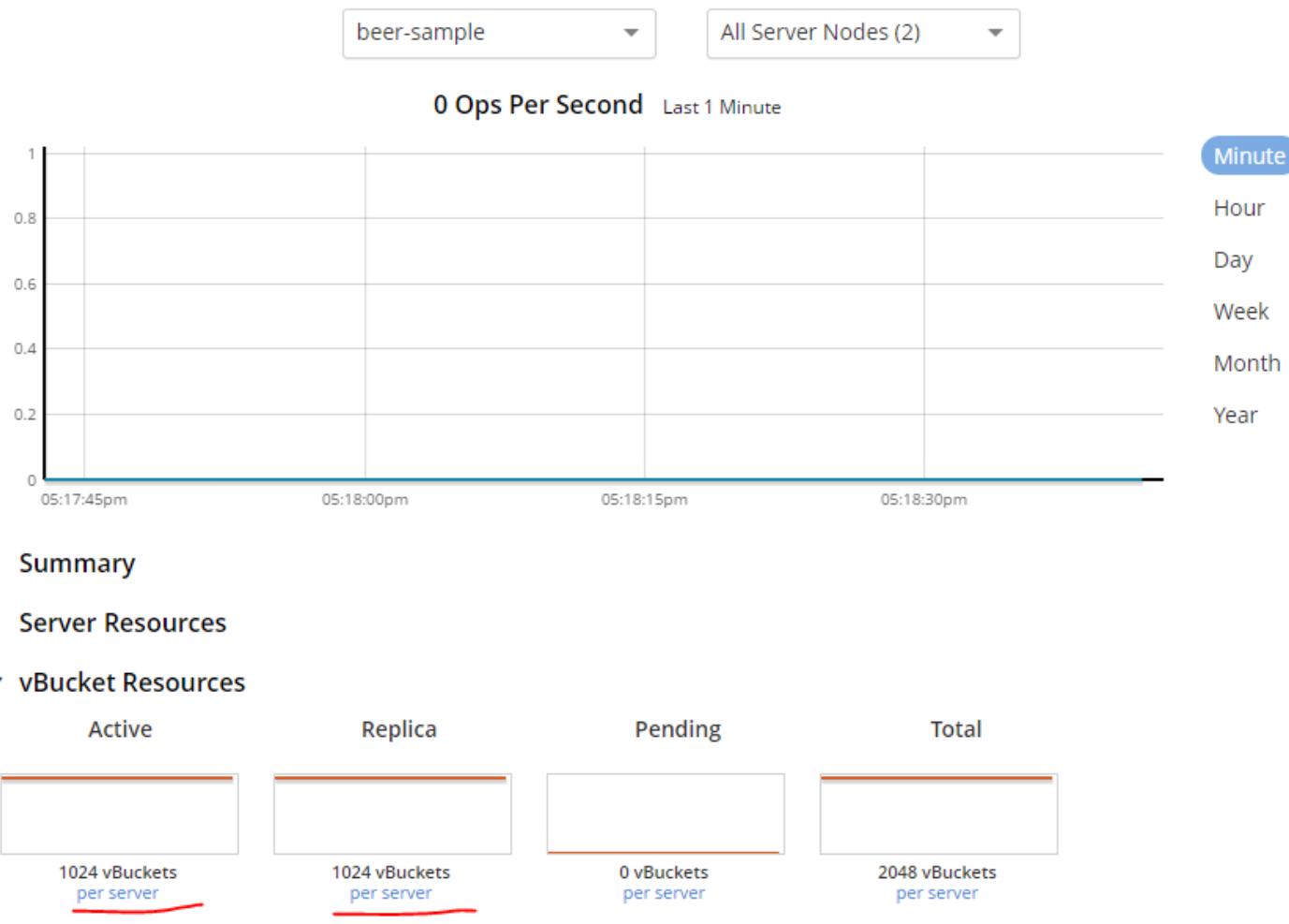
There are three types of failovers: *graceful*, *hard*, and *automatic*.

Use Graceful failover only when the following conditions are met; otherwise, use hard failover.  
All nodes in the cluster are healthy (for example, none has failed or is malfunctioning).

Each bucket in the cluster has all active vBuckets and at least one full set of replica vBuckets. If not, first fix the problems and rebalance the cluster. For each bucket, look at the vBucket Resources section of the metrics, then look at the active and replica vBucket graphs to confirm that each graph has 1024 for that bucket.

Bucket → beer-sample → Statistics

Or using Server -> Statistics. → All services → All nodes



Hard failover can be initiated in multiple ways:

- Manually via the Couchbase Web Console
- Using the CLI or RESTful API from a script/program/ops platform
- Using the automatic failover functionality performed by the Couchbase Cluster Manager

Automatic failover is:

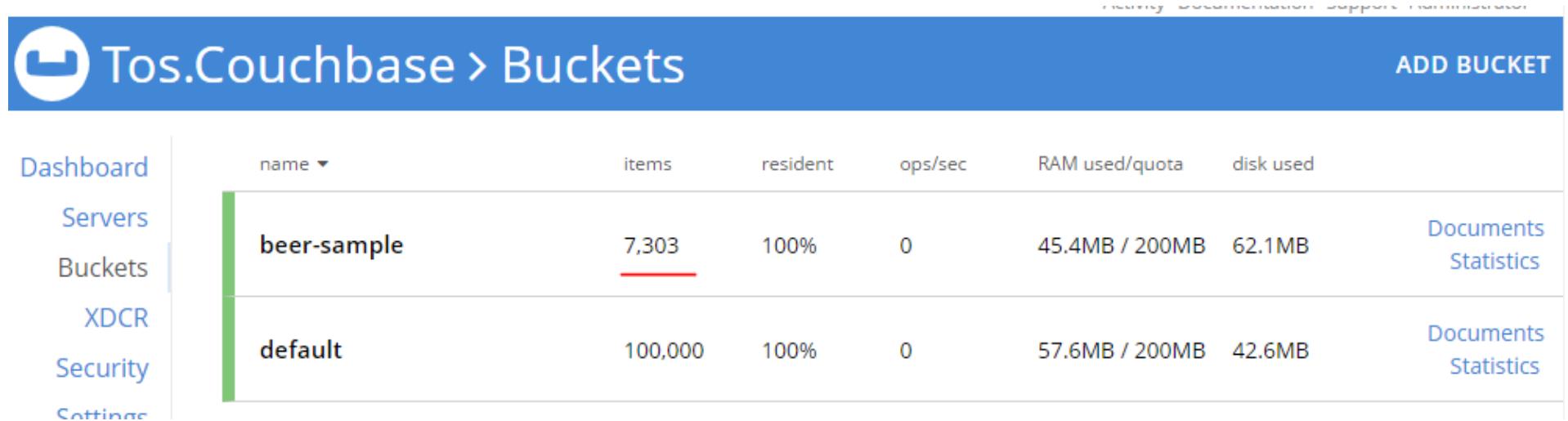
Disabled by default to prevent Couchbase Server from using it if you didn't enable it explicitly.

Available only on clusters that contain at least three nodes. This helps prevent a split-brain scenario in the cluster.

Designed to failover a node only if that node is the only one down at a given time. Combined with the previous restriction, this also prevents a split-brain scenario in the cluster.

## Single-node failure

First have a look at the buckets in your cluster. Note the number of items in the beer-sample bucket. You should see 7303 items (unless the sample bucket has changed).



The screenshot shows the Couchbase Web Console interface. The top navigation bar includes a logo, the text 'Tos.Couchbase > Buckets', and a 'ADD BUCKET' button. On the left, a vertical sidebar lists 'Dashboard', 'Servers', 'Buckets' (which is selected and highlighted in blue), 'XDCR', 'Security', and 'Settings'. The main content area is titled 'Buckets' and displays a table with the following data:

name ▾	items	resident	ops/sec	RAM used/quota	disk used	
beer-sample	7,303	100%	0	45.4MB / 200MB	62.1MB	<a href="#">Documents</a> <a href="#">Statistics</a>
default	100,000	100%	0	57.6MB / 200MB	42.6MB	<a href="#">Documents</a> <a href="#">Statistics</a>

The item count is an easy way to see how much data is potentially available.

Ok, now it's time to kill a node. Choose one of your Couchbase nodes (it doesn't matter which one) and either shut it down or just stop the Couchbase server service.

Let us do on master.

```
# service couchbase-server stop or systemctl stop couchbase-server
```

```
[root@tos ~]#  
[root@tos ~]# hostname  
tos.master.com  
[root@tos ~]# service couchbase-server stop  
Stopping couchbase-server  
[OK]  
[root@tos ~]#
```

If you were viewing the “failed” nodes web administration console you will be disconnected and should login to the other node’s web console.

You should see one node up and one down.

On Dashboard View

The screenshot shows the Tos.Couchbase dashboard with the following key metrics:

- Nodes:** 2 active nodes, 0 failed-over nodes, 0 nodes pending rebalance, 1 inactive node.
- Services:**
  - Data:** 2 nodes, 1 node not responding.
  - Index:** 1 node, 1 node not responding.
  - Search:** 1 node, 1 node not responding.
  - Query:** 1 node, 1 node not responding.
  - Eventing:** 1 node, 1 node not responding.
  - Analytics:** 1 node, 1 node not responding.
  - XDCR:** 0 remote clusters, 0 replications.

## On Servers view

A warning message is displayed at the top: "Warning: Additional active servers required to provide the desired number of replicas."

The Servers view table includes the following columns: name, group, services, CPU, RAM, swap, disk used, and items.

name	group	services	CPU	RAM	swap	disk used	items
192.168.139.129	Group 1	analytics data eventing index query search	31.3%	28.6%	0%	58.5MB	53.7 K...
tos.slave.com	Group 1	data	5.12%	53.9%	2.23%	33.3MB	53.5 K... <a href="#">Statisti...</a>

A message below the table states: "Node unresponsive | Not taking traffic | FAILOVER to activate available replicas" with a "Failover" button.

Now have a look at your buckets. Note that the item count is now reduced by 50%. The data is still safe because the data was replicated and evenly distributed on all nodes. We are seeing a reduced item count because half the active data is gone.

The screenshot shows the Couchbase Bucket Management interface. On the left, a sidebar menu includes Dashboard, Servers, **Buckets**, XDCR, Security, Settings, Logs, and Documents. The main area displays two buckets:

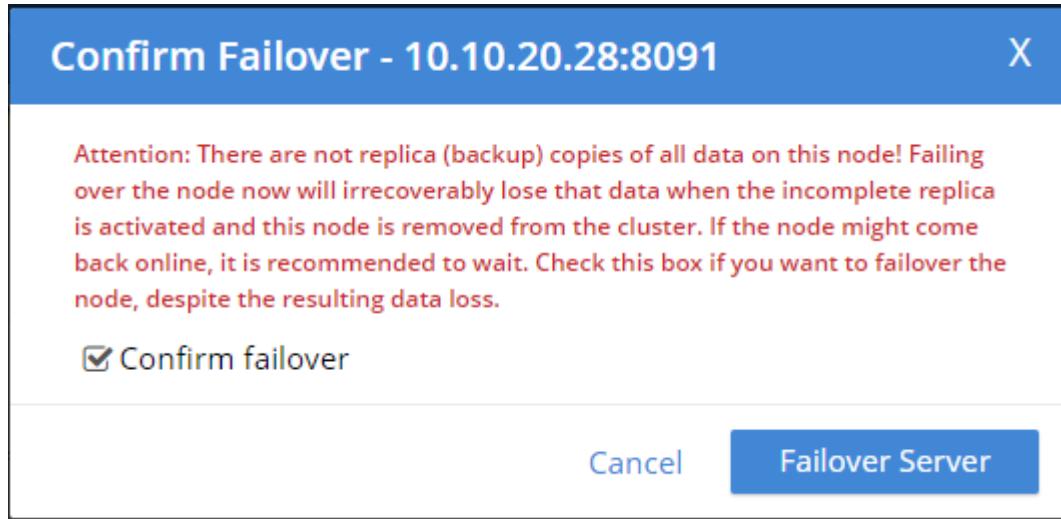
name ▾	items	resident	ops/sec	RAM used/quota	disk used	
beer-sample	3,592	100%	0	24.7MB / 200MB	15.7MB	<a href="#">Documents</a> <a href="#">Statistics</a>
	1 node not responding					
default	50,000	100%	0	31.8MB / 200MB	17.5MB	<a href="#">Documents</a> <a href="#">Statistics</a>
	1 node not responding					

To get back access to all of our data we need to make the replica data (on our remaining node) active. This is actually really easy. Just click “Fail Over” on the down node. **Hard Failover** since the server node is down.

name	group	services	CPU	RAM	swap	disk used	items	Rebalance
10.10.20.28	Group 1	data full text index query	0%	---	---	99.7MB	65.7 K/65.8 K	<b>Failover</b>
tos.slave.com	Group 1	data	4.12%	39.4%	0%	51.4MB	65.8 K/65.7 K	Statistics

You will be presented with the very scary data loss warning. In some circumstances you will lose data but not with this simple scenario.

The “down” server will be added to the “pending rebalance” tab. If you rebalance now, any data not replicated across the cluster on the “down” server will be lost. If the “down” server comes back online while it is pending rebalance you will be prompted to add the server back. If you did rebalance, the server will have to be reconfigured manually to join the cluster again.



Click on Failover Server

Then

Click on rebalance

Rebalance

Have a look at your buckets now. Item count should be increased and it should look the same as before, except you now only have 1 node. i.e slave

## Bucket View

The screenshot shows the Couchbase Bucket View. On the left, there is a vertical navigation bar with links: Dashboard, Servers, Buckets, XDCR, Security, and Settings. The 'Buckets' link is highlighted. The main area displays a table with the following data:

	name ▾	items	resident	ops/sec	RAM used/quota	disk used	
	<b>beer-sample</b>	7,303	100%	0	24.9MB / 100MB	17.7MB	<a href="#">Documents</a> <a href="#">Statistics</a>
	<b>default</b>	100,000	100%	0	31.9MB / 100MB	19.5MB	<a href="#">Documents</a> <a href="#">Statistics</a>

## Servers View.

The screenshot shows the Couchbase Servers View. On the left, there is a vertical navigation bar with links: Dashboard, Servers, Buckets, and Indexes. The 'Servers' link is highlighted. The main area displays a table with the following data:

	name ▾	group	services	CPU	RAM	swap	disk used	items	Rebalance
	<b>tos.slave.com</b>	Group 1	data	1.51%	38.1%	0%	55.4MB	131 K/0	<a href="#">Statistics</a>

Your Couchbase cluster should now be working (but slower and without replication).

Uninstall the couchbase from the master node and re install it to add back to the cluster.

Steps: - On the Master Node

stop the couchbas-server : systemctl stop couchbase-server.

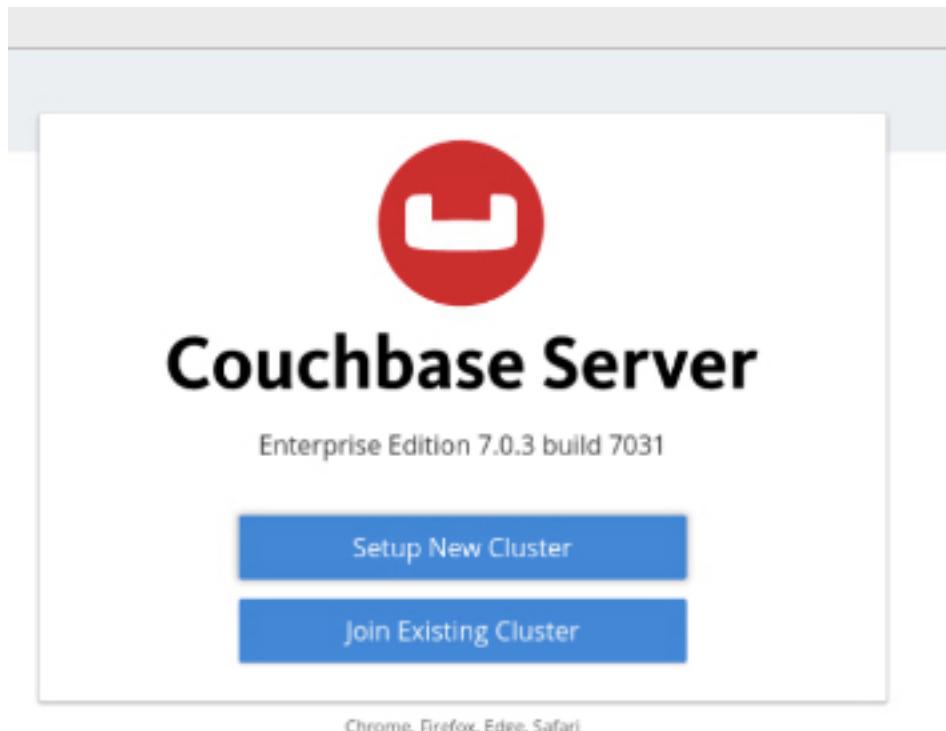
Uninstall : rpm --erase couchbase-server

Install : rpm --install couchbase\*rpm

Next we will **Add the node back and rebalance it.**

Once complete your cluster should be running with 2 nodes.

If you access the web console of the master it will be in the following state.



Go to the slave Node and Perform the following steps to Add the master back in the cluster.

Click on Servers → Add Server using the web console of the Slave Node. Hostname will be tos.hp.com or your server hostname .i.e the one which was down earlier.

## Add Server Node

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

**Hostname/IP Address**

**Username** an existing username with admin access to this server

**Password** an existing password with admin access to this server

**Services** ⓘ

- Data
- Index
- Search
- Query
- Eventing
- Analytics

[Cancel](#) [Add Server](#)

Click Add Server

The **GSI Index and Full Text and Analytics and Eventing** services have been added to the cluster for the first time. Please review and set the following service-specific settings.

#### RAM Quotas in megabytes per server node

Data	<input type="text" value="256"/> MB
Index	<input type="text" value="256"/> MB
Search	<input type="text" value="256"/> MB
Query	-----
Eventing	<input type="text" value="256"/> MB
<b>The eventing service quota (120MB) cannot be less than 256MB.</b>	
Analytics	<input type="text" value="1024"/> MB

RAM Available 1831MB Max Allowed Quota 1465MB

#### Index Storage Setting

- Standard Global Secondary
- Memory-Optimized (i)

**Save Settings**

Click on Save Settings.

Let us rebalance the bucket for that Click on rebalance:

name	group	services	CPU	RAM	swap	disk used	items	
<b>tos.master.com</b>	Group 1	data full text index query	1.51%	25%	0%	---	0/0	<a href="#">Statistics</a>
<b>New node   Not taking traffic   ADD pending rebalance</b>								<a href="#">Cancel Add</a>
<b>tos.slave.com</b>	Group 1	data	4.08%	37.7%	0%	55.4MB	131 K/0	<a href="#">Statistics</a>

name	group	services	CPU	RAM	swap	disk used	items	
<b>tos.master.com</b>	Group 1	data full text index query	83.5%	33%	0%	34B	644/0	<a href="#">Statistics</a>
<b>tos.slave.com</b>	Group 1	data	5.1%	37.5%	0%	55.4MB	131 K/0	<a href="#">Statistics</a>

All documents should be distributed and intact now. After sometimes, Node status should be as shown below:

Nodes								Failover Multiple Nodes	Rebalance
name	group	services	CPU	RAM	swap	disk used	items		
tos.master.com	Group 1	analytics data eventing index query search	11.8%	50.2%	0%	41.4MB	53.7 K...	Statistics	
tos.slave.com	Group 1	data	3.62%	53.8%	0%	58.3MB	53.5 K...	Statistics	

And bucket status should be:

name	items	resident	ops/sec	RAM used/quota	disk used
default	100,000	100%	0	68.6MB / 200MB	46.8MB
travel-sample	31,591	100%	0	128MB / 200MB	80.4MB

## Recover a Node and Rebalance

After a node has been failed over, it can be *recovered*: that is, added back into the cluster from which it was failed over, by means of the *rebalance* operation.

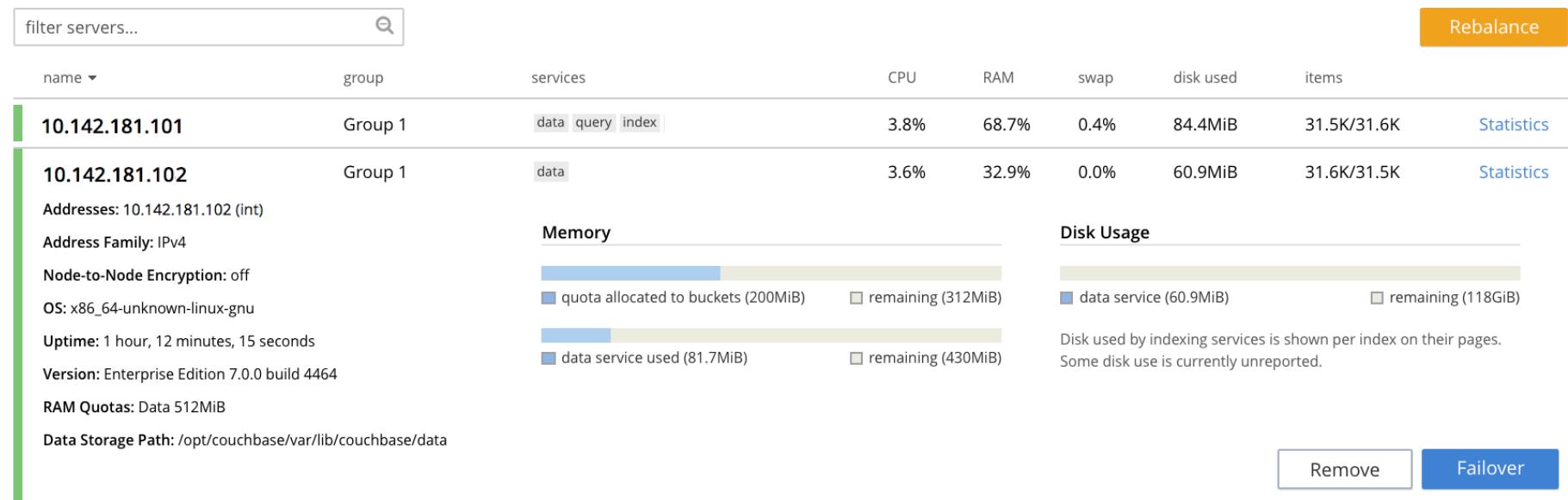
After failover has occurred, before the failed over node has been rebalanced out of the cluster, the node can be recovered, and thereby reintegrated into the cluster. This is useful in circumstances where, following failover, the unhealthy node has been fixed, and is therefore now assumed fit for re-addition.

You can bring down a node using **Failover** option, however the node is still part of the cluster.

## Recover a Node with the UI

Proceed as follows:

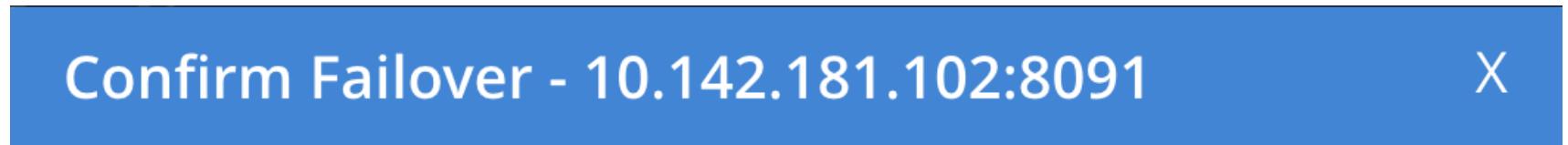
- 13 Access the Couchbase Web Console **Servers** screen, on node **master**, by left-clicking on the **Servers** tab in the left-hand navigation bar.
- 14 To see further details of each node, left-click on the row for the node. The row expands vertically, as follows: Use **slave**



- 15 To initiate failover, left-click on the **Failover** button, at the lower right of the row for **slave**:



The **Confirm Failover Dialog** now appears:



## Failover Options

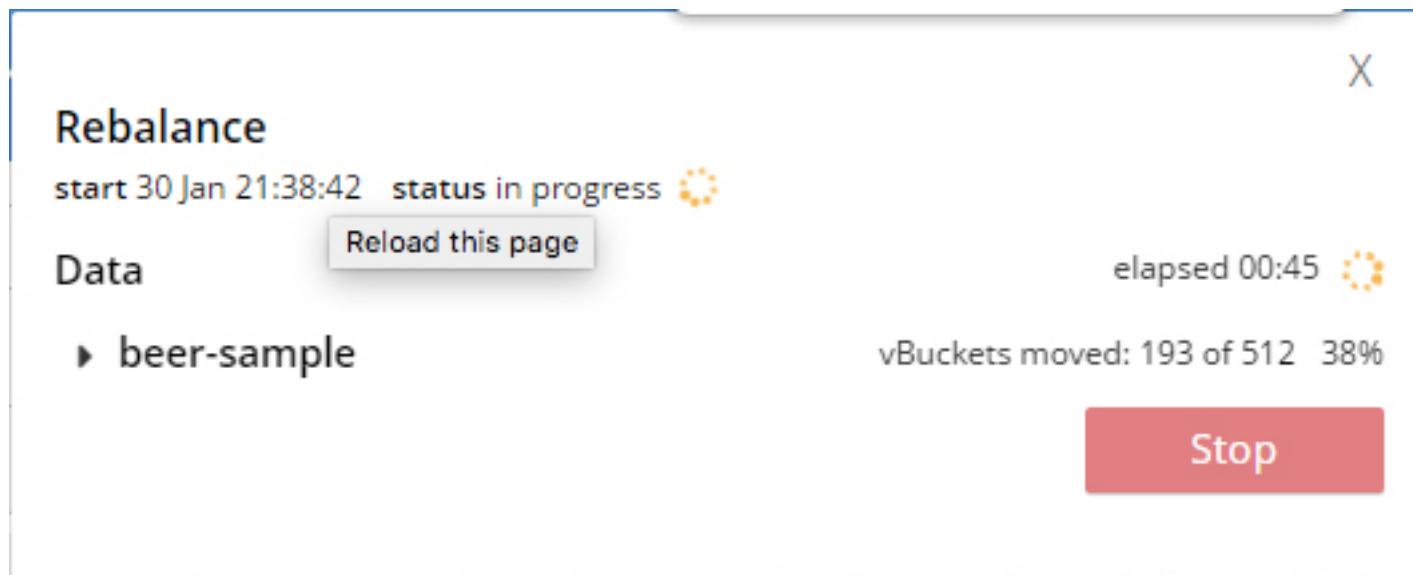
- Graceful Failover (default)
- Hard Failover

Cancel

**Failover Node**

Two radio buttons are provided, to allow selection of either **Graceful** or **Hard** failover. **Graceful** is selected by default.

- 16 Confirm *graceful* failover by left-clicking on the **Failover Node** button. Graceful failover is now initiated. A progress dialog appears new the top of the screen, summarizing overall progress.



When the process ends, the display is as follows:

filter servers... 

**Rebalance**

name ▾	group	services	CPU	RAM	swap	disk used	items	
10.142.181.101	Group 1	data query index	99.4%	76.9%	3.6%	70.8MiB	25.7K/21.6K	<a href="#">Statistics</a>
10.142.181.102	Group 1	data	0.0%	---	---	---	0/0	

**Node failed-over | Not available for traffic | REBALANCE to finish removing node**

This server is now reachable. Do you want to add it back to the cluster on the next rebalance?

[Add Back: Full Recovery](#) [Add Back: Delta Recovery](#)

Addresses: 10.142.181.102 (int)  
Address Family: IPv4  
Node-to-Node Encryption: off  
OS: x86\_64-unknown-linux-gnu  
Uptime: 48 minutes, 8 seconds  
Version: Enterprise Edition 7.0.0 build 4464  
RAM Quotas: Data 512MiB  
Data Storage Path:  
/opt/couchbase/var/lib/couchbase/data

**Memory**

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

data service used (0B)    remaining (512MiB)

**Disk Usage**

data service (0B)    remaining (118GiB)

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

This indicates that the graceful failover has successfully completed.

This way you can take a node for maintenance and add back to cluster without removal.

You can verify any bucket and all document should be intact and readable.

Warning: At least two servers with the data.

	21st_amendment_brewery_cafe-563_stout	21st_amendment_brewery_cafe-
	{ "name": "563 Stout", "abv": 5.0, "ibu": 0.0, "srn": 0.0, "upc": 0, "type": "beer", "brewery_id": "21", "date": "2010-07-22 20:00:20", "description": "Deep black color, toasted black b..." }	{ "name": "Amendment Pale Ale", "abv": 5.0, "ibu": 0.0, "srn": 0.0, "upc": 0, "type": "beer", "brewery_id": "21", "date": "2010-07-22 20:00:20", "description": "Deep golden color. Citrus and piney..." }

Let us add back the Node into the cluster.

A rebalance is required to complete the reduction of the cluster to one node. Additionally, the **Add Back: Full Recovery** and **Add Back: Delta Recovery** buttons are displayed, towards the left-hand side of the row - **slave**:

Add Back: Full Recovery

Add Back: Delta Recovery

- 17 Select one of the two available forms of recovery, by left-clicking the corresponding button. Note that *full* and *delta* recovery are described in [Recovery](#). If you select *full*, by left-clicking on the **Add Back: Full Recovery** button, the row for **slave** is displayed as follows:

name	group	services	CPU	RAM	swap	disk used	items	
10.142.181.101	Group 1	data query index	0.0%	---	---	---	63.1K/0	<a href="#">Statistics</a>
10.142.181.102	Group 1	data	2.1%	28.6%	0.0%	---	0/0	<a href="#">Statistics</a>

Node failed-over | Not available for traffic | REBALANCE to finish full recovery

Cancel Add Back

Addresses: 10.142.181.102 (int)  
Address Family: IPv4  
Node-to-Node Encryption: off  
OS: x86\_64-unknown-linux-gnu  
Uptime: 54 minutes, 21 seconds  
Version: Enterprise Edition 7.0.0 build 4464  
RAM Quotas: Data 512MiB  
Data Storage Path: /opt/couchbase/var/lib/couchbase/data

Memory

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

Disk Usage

- data service (0B)
- remaining (118GiB)

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

The row specifies REBALANCE to finish full recovery: therefore, left-click the **Rebalance** button to apply full recovery. Similarly, left-clicking on the **Add Back: Delta Recovery** displays REBALANCE to finish delta recovery. Recovery can be aborted, by left-clicking on the **CANCEL ADD BACK** button.

Left-click on the **Rebalance** button. Whichever form of recovery you have chosen, *full* or *delta*, is performed.

## Removing a Node – Gracefully

Before you remove a node from the cluster, ensure that you have the capacity within the remaining nodes to handle the workload. Removing a node does not stop the node from servicing requests. Instead, it only marks the node as ready for removal from the cluster. You must perform a rebalance operation to complete the removal process. Once a node is removed, it is no longer part of the cluster and can be switched off, updated, or upgraded.

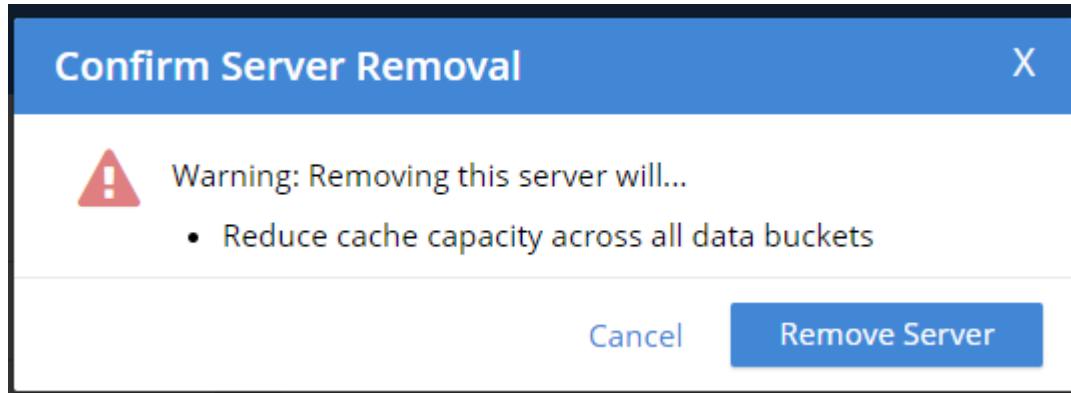
### Remove Nodes via UI

Remove a node from the cluster from within the Server Nodes section of the Couchbase Web Console. Let us remove the slave Node. Servers → tos.slave → Remove.

The screenshot shows the Couchbase Web Console interface under the 'Servers' tab. On the left, a sidebar lists various management sections: Dashboard, Servers (selected), Buckets, Indexes, Search, Query, XDCR, Security, Settings, and Logs. The main content area displays a table of server details. The 'tos.master.com' row shows services: data, full text, index, query; CPU: 6.18%; RAM: 42.8%; swap: 0%; disk used: 74.1MB; items: 65.7 K/65.8 K. The 'tos.slave.com' row shows services: data; CPU: 7.73%; RAM: 36.6%; swap: 0%; disk used: 87.5MB; items: 65.8 K/65.7 K. Below the table, detailed information for 'tos.slave.com' includes Uptime (6 minutes, 45 seconds), OS (x86\_64-unknown-linux-gnu), Version (Enterprise Edition 5.0.1 build 5003), Data Service RAM Quota (512 MB), Data Storage Path (/opt/couchbase/var/lib/couchbase/data), and Index Storage Path (/opt/couchbase/var/lib/couchbase/data). At the bottom right of the table are two buttons: 'Rebalance' (orange) and 'Remove' (blue).

Click **Remove** to mark the node for removal.

You will be prompted with a warning message as shown below:

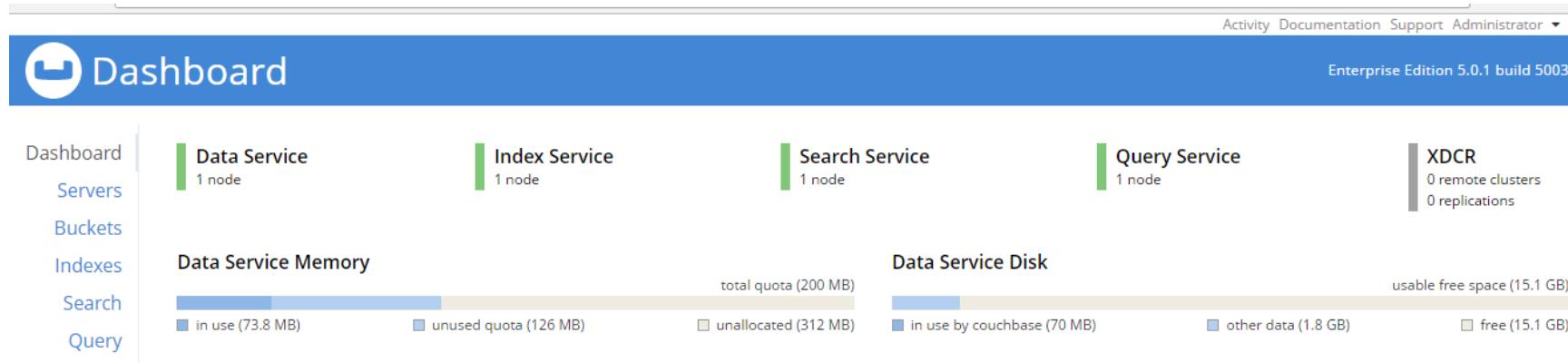


Click Remove

After the node is removed, Click on **Rebalance**.

A screenshot of the Cluster Operations dashboard. A modal window in the center says 'rebalancing 2 nodes 8.9%' with a progress bar. To the right is a 'Stop Rebalance' button. Below the modal is a table with columns: name, group, services, CPU, RAM, swap, disk used, items. One row shows 'tos.master.com Group 1' with services 'data full text index query' and usage statistics: 92% CPU, 46.4% RAM, 0% swap, 74.4MB disk used, 72.8 K/58.5 K items. A progress bar at the bottom indicates '1.8% complete'. The top navigation bar includes 'Activity 1', 'Documentation', 'Support', 'Administrator', 'FILTER', and a 'Stop Rebalance' button.

Then you can safely shutdown the slave node to reduce the cluster capacity after it's fully rebalance



In this lab:

We have learned the following:

- Add Node to a cluster ( Use when you want to increase the cluster capacity)
- Failover of Node (Use when you will bring the node later after system maintenance)
- Removing Node (use when you want to reduce the cluster capacity)

-----End of Lab-----

## 11. View Creation – 30 Minutes

As a system administrator, you will be able to create views using Views editor at the end of this lab.  
Log on the web console using the Administrator credentials.

Click on Indexes – Views -> (Beer-sample bucket) → Development views -> Add View

Ensure to select the beer-sample bucket.

In the Add View Enter the following details.

Design Document Name : brewery

View Name : b-city

Click Save, to save the View.

You will be able to see the view from the view console.

To add the Map code you can click on the Edit option and paste the following in the Map pane.

```
function(doc, meta) {
  if (doc.type == "brewery" && doc.city) {
    emit(doc.city,null);
}
```

{

Update the view as follows:

[View Index Code](#)

Map

```
1 function(doc, meta) {  
2     if (doc.type == "brewery" && doc.city) {  
3         emit(doc.city,null);  
4     }  
5 }  
6 }
```

Click on Save Changes which is on the right above of the Reduce Pane.

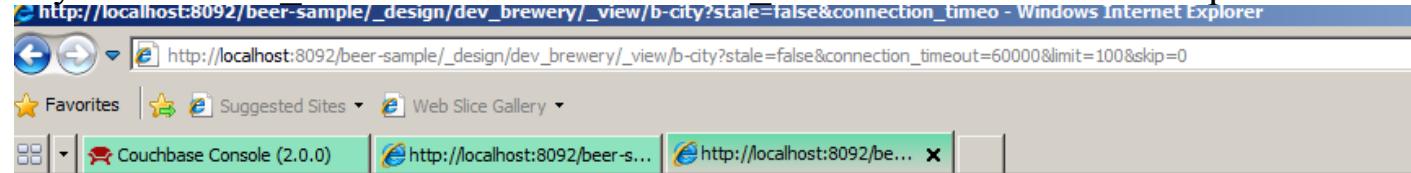
Click on show results to determine the data output of the View.

	Key	Value
"Aberdeen"	<a href="#">pinehurst_village_brewery</a>	null
"Abingdon"	<a href="#">morland_and_co</a>	null
"Abita Springs"	<a href="#">abita_brewing_company</a>	null
"Achel"	<a href="#">brouwerij_de_achelse_kluis</a>	null
"Achel"	<a href="#">brouwerij_der_sint_benedictusabdij_de_achelse_kluis</a>	null
"Adamstown"	<a href="#">stoudt_s_brewery</a>	null

You may have observed that values are null. Why? [Hints → data can be fetched using Document key.]

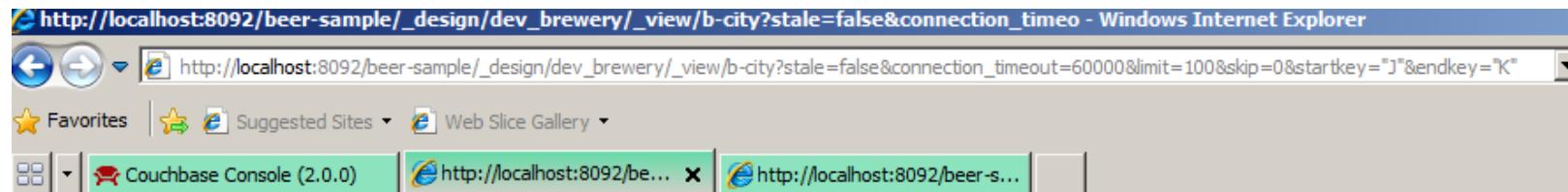
Ensure that the host name is able to resolve from your client machine. You can call the following URL to call this particular view. Enter the Administrator credentials when prompt for the user credentials.

`http://tos.hp.com:8092/beer-sample/_design/dev_brewery/_view/b-city?connection_timeout=60000&inclusive_end=true&limit=600&skip=0&stale=false`



```
{"total_rows":1390,"rows": [
{"id":"pinehurst_village_brewery","key":"Aberdeen","value":null},
 {"id":"morland_and_co","key":"Abingdon","value":null},
 {"id":"abita_brewing_company","key":"Abita Springs","value":null},
 {"id":"brouwerij_de_achelse_kluis","key":"Achel","value":null},
 {"id":"brouwerij_der_sint_benedictusabdij_de_achelse_kluis","key":"Achel","value":null},
 {"id":"stoudt_s_brewery","key":"Adamstown","value":null},
 {"id":"hoffbrau_steaks_brewery_1","key":"Addison","value":null},
 {"id":"south_australian_brewing","key":"Adelaide","value":null},
 {"id":"brasserie_des_cimes","key":"Aix les Bains","value":null},
 {"id":"hoppin_frog_brewery","key":"Akron","value":null},
 {"id":"thirsty_dog_brewing","key":"Akron","value":null},
 {"id":"c_h_evans_brewing_company","key":"Albany","value":null},
 {"id":"oregon_trader_brewing","key":"Albany","value":null},
 {"id":"blue_corn_caf_and_brewery_albuquerque","key":"Albuquerque","value":null},
 {"id":"chama_river_brewing","key":"Albuquerque","value":null},
 {"id":"dry_gulch_brewing","key":"Albuquerque","value":null},
 {"id":"marble_brewery","key":"Albuquerque","value":null},
 {"id":"founder_s_restaurant_brewing","key":"Alexandria","value":null},
 {"id":"allentown_brew_works","key":"Allentown","value":null},
 {"id":"williams_brothers_brewing_company","key":"Alloa","value":null},
 {"id":"alpine_beer_company","key":"Alpine","value":null},
 {"id":"alpirsbacher_klosterbru","key":"Alpirsbach","value":null},
 {"id":"harviestoun_brewery","key":"Alva","value":null},
 {"id":"millstream_brewing","key":"Amana","value":null},
 {"id":"olde_main_brewing","key":"Ames","value":null},
 {"id":"amherst_brewing_company","key":"Amherst","value":null},
 {"id":"central_waters_brewing_company","key":"Amherst","value":null},
 {"id":"amstel_brouwerij","key":"Amsterdam","value":null},
 {"id":"brouwerij_t_ij","key":"Amsterdam","value":null},
 {"id":"anacortes_brewing","key":"Anacortes","value":null},
```

Append - &startkey="J"&endkey="K" in the above URL. It will filters for the key between J and K alphabet.



```

http://localhost:8092/beer-sample/_design/dev_brewery/_view/b-city?stale=false&connection_timeout=60000&limit=100&skip=0&startkey="J"&endkey="K"

{
  "total_rows": 1390,
  "rows": [
    {"id": "snake_river_brewing", "key": "Jackson", "value": null},
    {"id": "margaritaville_brewing_company", "key": "Jacksonville", "value": null},
    {"id": "brasserie_de_vervifontaine", "key": "Jalhay-Vervifontaine", "value": null},
    {"id": "gray_brewing", "key": "Janesville", "value": null},
    {"id": "folx_les_caves", "key": "Jauche", "value": null},
    {"id": "brasserie_duyck", "key": "Jenlain", "value": null},
    {"id": "friesisches_brauhaus_zu_jever", "key": "Jever", "value": null},
    {"id": "jever_brewery", "key": "Jever, Germany", "value": null},
    {"id": "brouwerij_alken_maes", "key": "Jumet", "value": null},
    {"id": "alaskan_brewing", "key": "Juneau", "value": null}
  ]
}

```

Create the following View in the Beer sample Bucket:

Design Document Name : brewery

Name : Specific\_Attribute

```

function (doc, meta) {
  if (doc.type === "beer" && doc.brewery_id) {
    emit(doc.brewery_id, doc.abv);
  }
}

```

▼ VIEW CODE

Map

```
1 function (doc, meta) {
2 if(doc.type == "beer" && doc.brewery_id) {
3   emit(doc.brewery_id,doc.abv);
4 }
5 }
```

## Show result

Results  Development Time Subset Full Cluster Data Set

Key	Value
"21st_amendment_brewery_cafe"	7.2
<a href="#">21st_amendment_brewery_cafe-21a_ipa</a>	
"21st_amendment_brewery_cafe"	5
<a href="#">21st_amendment_brewery_cafe-563_stout</a>	
"21st_amendment_brewery_cafe"	5.2
<a href="#">21st_amendment_brewery_cafe-amendment_pale_ale</a>	

Update the Specific\_Attribute views as below:  
Add \_stats reduce function by clicking on it.

[View Index Code](#)

Map

```
1 function (doc, meta) {
2   if (doc.type == "beer" && doc.brewery_id) {
3     emit(doc.brewery_id, doc.abv);
4   }
5 }
```

[Make Copy](#)

[Save Changes](#)

Reduce (built in: \_count, \_sum, \_stats)

```
1 _stats
```

[http://tos.hp.com:8092/beer-sample/\\_design/dev\\_brewery/\\_view/Specific\\_Attribute?stale=false&connection\\_timeout=60000&limit=10&skip=0](http://tos.hp.com:8092/beer-sample/_design/dev_brewery/_view/Specific_Attribute?stale=false&connection_timeout=60000&limit=10&skip=0)

← → ⌂ ⓘ Not secure | tos.hp.com:8092/beer-sample/\_design/dev\_brewery/\_view/Specific\_Attribute?stale=false&connection\_timeout=60000&limit=10&skip=0

```
{"total_rows":5891,"rows": [
{"id":"21st_amendment_brewery_cafe-21a_ipa","key":"21st_amendment_brewery_cafe","value":7.2},
 {"id":"21st_amendment_brewery_cafe-563_stout","key":"21st_amendment_brewery_cafe","value":5},
 {"id":"21st_amendment_brewery_cafe-amendment_pale_ale","key":"21st_amendment_brewery_cafe","value":5.2},
 {"id":"21st_amendment_brewery_cafe-bitter_american","key":"21st_amendment_brewery_cafe","value":3.6},
 {"id":"21st_amendment_brewery_cafe-double_trouble_ipa","key":"21st_amendment_brewery_cafe","value":9.8},
 {"id":"21st_amendment_brewery_cafe-general_pippo_s_porter","key":"21st_amendment_brewery_cafe","value":5.5},
 {"id":"21st_amendment_brewery_cafe-north_star_red","key":"21st_amendment_brewery_cafe","value":5.8},
 {"id":"21st_amendment_brewery_cafe-oyster_point_oyster_stout","key":"21st_amendment_brewery_cafe","value":5.9},
 {"id":"21st_amendment_brewery_cafe-potrero_esb","key":"21st_amendment_brewery_cafe","value":5.2},
 {"id":"21st_amendment_brewery_cafe-south_park_blonde","key":"21st_amendment_brewery_cafe","value":5}
]}
```

**Append in URL:** - &group\_level=1

← → ⌂ ⓘ Not secure | tos.hp.com:8092/beer-sample/\_design/dev\_brewery/\_view/Specific\_Attribute?stale=false&connection\_timeout=60000&limit=10&sl

```
{"rows": [
{"key": "21st_amendment_brewery_cafe", "value": {"sum": 63.7, "count": 11, "min": 3.6, "max": 9.800000000000001, "sumsqr": 393.8700000000001}, 
 {"key": "3_fonteinen_brouwerij_ambachtelijke_geuzestekerij", "value": {"sum": 11, "count": 2, "min": 5, "max": 6, "sumsqr": 61}}, 
 {"key": "512_brewing_company", "value": {"sum": 54.6, "count": 8, "min": 5.2, "max": 8.199999999999999, "sumsqr": 380.92}}, 
 {"key": "aass_brewery", "value": {"sum": 22.8, "count": 5, "min": 0, "max": 5.9, "sumsqr": 130.12}}, 
 {"key": "abbaye_de_leffe", "value": {"sum": 13.4, "count": 2, "min": 6.6, "max": 6.8, "sumsqr": 89.799999999998}}, 
 {"key": "abbaye_de_maredsous", "value": {"sum": 24, "count": 3, "min": 6, "max": 10, "sumsqr": 200}}, 
 {"key": "abbaye_notre_dame_du_st_remy", "value": {"sum": 28, "count": 3, "min": 7.5, "max": 11.3, "sumsqr": 268.58}}, 
 {"key": "aberdeen_brewing", "value": {"sum": 4.8, "count": 1, "min": 4.8, "max": 4.8, "sumsqr": 23.04}}, 
 {"key": "abhi_brewery", "value": {"sum": 0, "count": 1, "min": 0, "max": 0, "sumsqr": 0}}, 
 {"key": "abita_brewing_company", "value": {"sum": 34.85, "count": 20, "min": 0, "max": 8, "sumsqr": 210.0725}}
]
}
```

-----End of Lab-----

## 12. Query Workbench & Index- 35 Minutes

In this lab, we will understand an overview of Query Workbench in Couchbase cluster

The Query Workbench provides a rich graphical user interface to perform query development.

The Query Workbench consists of three working areas as shown in the following figure:

1. N1QL Editor
2. Bucket Analysis
3. Results

The screenshot shows the Couchbase Query Workbench interface. At the top, there are two tabs: "Query Workbench" (selected) and "Query Monitor".

**Query Editor:** This section contains the N1QL query code:

```
1 SELECT p.owner.username, p.name AS trackname,
2 {"artist": t.artist, "title": t.title, "url": t.mp3 } AS trackdetail
3 FROM couchmusic2 AS p
4 USE KEYS "playlist::00011b74-12be-4e60-abbf-b1c8b9b40bfe" JOIN couchmusic2 AS t
5 ON KEYS ARRAY "track:::" || trackId FOR trackId IN p.tracks END;
```

Below the query editor are two buttons: "Execute" (highlighted in blue) and "Explain". To the right of these buttons are links for "cached query", "elapsed:", "execution:", "count:", and "size:". There is also a "Preferences" link.

**Bucket Insights:** This section displays information about buckets:

- Fully Queryable Buckets:**
  - beer-sample (7305)
  - couchmusic2 (213063)
- Queryable on Indexed Fields**
- Non-Indexed Buckets:**
  - default (1)

**Query Results:** This section shows the results of the executed query. It includes tabs for "JSON", "Table" (selected), "Tree", "Plan", and "Plan Text". The results table is currently empty and displays the message "data\_not\_cached". Below the table, there is a note: "Hit execute to rerun query".

Click on Query -->

You need to create an index before executing the command.

SELECT name, brewery\_id from `beer-sample` WHERE brewery\_id IS NOT MISSING LIMIT 2;

The screenshot shows the Couchbase Query Workbench. At the top, there is a toolbar with buttons for 'Execute', 'Clear History', and 'Save Query'. Below the toolbar, a query editor window contains the following code:

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

On the left side of the interface, there is a 'Bucket Analysis' panel. It lists 'Fully Queryable Buckets', 'Queryable on Indexed Fields', and 'Non-Indexed Buckets'. Under 'Non-Indexed Buckets', it shows a section for 'beer-sample' with the following details:

- Summary: 2 flavors found, s
- Flavor 1 ( 19.9% ), in-comm
- address (array)
- city (string)
- code (string)
- country (string)
- description (string)
- geo (object), child type:
- accuracy (string)

The main workspace displays the results of the query execution. The status bar at the top of the results area shows: Status: 404, Elapsed: 1.04ms, Execution: 910.672µs, Result Count: 0, and Result Size: 0. The results pane itself is empty, indicating no data was returned due to the missing index error.

You will get the above error if primary index is not created before executing any queries.

Before you can run your first query, create a primary index on the bucket, for example `beer-sample`. The primary index contains a list of every document within the database, with the document ID as the key.

`CREATE PRIMARY INDEX ON `beer-sample` USING GSI;`

The screenshot shows the Couchbase Query Workbench interface. At the top, there is a toolbar with buttons for 'Execute', 'Clear History', and 'Save Query'. Below the toolbar, a code editor window displays the command: `CREATE PRIMARY INDEX ON `beer-sample` USING GSI;`. To the right of the code editor is a results panel. The results panel has tabs for 'JSON', 'Table', and 'Tree'. The 'JSON' tab is selected, showing the following JSON output:

```

{
  "status": "success",
  "elapsed": 5.22s,
  "execution": 5.22s,
  "resultCount": 0,
  "resultSize": 0
}

```

On the left side of the interface, there is a 'Bucket Analysis' sidebar. It lists 'Fully Queryable Buckets' and 'Queryable on Indexed Fields'. Under 'Non-indexed Buckets', it shows a list starting with 'beer-sample'. A summary message for 'beer-sample' states: 'Summary: 2 flavors found, s Flavor 1 ( 19.9%) , in-comm address (array) city (string) code (string) country (string) description (string) geo (object), child type: accuracy (string)'.

Remember that for all identifiers (bucket names) that contain a hyphen character you need to enclose the name with backtick (`) characters.

Now that your bucket is indexed, you can run some queries to explore the data it contains. Here are few sample queries to get you started.

The following query returns the different values used for the type field:

`SELECT DISTINCT type FROM `beer-sample`;`

The screenshot shows the Couchbase Query Workbench interface. At the top, there are buttons for 'Execute' (highlighted in orange), 'Clear History', and 'Save Query'. Below the buttons, the query 'SELECT DISTINCT type FROM `beer-sample`;' is entered. The results section shows the following details:

Status:	success
Elapsed:	1.03s
Execution:	1.03s
Result Count:	2
Result Size:	79

The results are displayed in JSON format:

```
1 [ ]  
2 {  
3   "type": "beer"  
4 },  
5 {  
6   "type": "brewery"  
7 }  
8 ]
```

On the left side, there is a 'Bucket Analysis' panel with the following summary:  
Queryable Buckets: beer-sample  
Queryable on Indexed Fields: beer-sample  
n-Indexed Buckets: beer-sample  
Summary: 2 flavors found, s  
Flavor 1 ( 19.9% ) , in-comm  
address (array)  
city (string)  
code (string)  
country (string)  
description (string)  
geo (object). child type:

```
SELECT name FROM `beer-sample` WHERE brewery_id ="mishawaka_brewing";
```

The screenshot shows the Couchbase Query Workbench interface. At the top, there's a toolbar with 'Execute' (highlighted in orange), 'Clear History', and 'Save Query'. Below the toolbar, a query is entered:

```
1 SELECT name FROM `beer-sample` WHERE brewery_id = "mishawaka_brewing";
```

The main area is divided into sections: 'Bucket Analysis' (which lists 'beer-sample' bucket details like fields and types), 'Results' (which displays the query status, elapsed time, execution time, result count, and result size), and a large JSON results pane.

**Bucket Analysis:**

- Fully Queryable Buckets
- Queryable on Indexed Fields
- Non-Indexed Buckets
- beer-sample
  - Summary: 2 flavors found, s
  - Flavor 1 (19.9%), in-comm
  - address (array)
  - city (string)
  - code (string)
  - country (string)
  - description (string)
  - geo (object), child type:
  - accuracy (string)
  - lat (number)
  - lon (number)
  - name (string)
  - phone (string)
  - state (string)
  - type (string)
  - undefined (string)

**Results:**

Status: success	Elapsed: 1.05s	Execution: 1.05s	Result Count: 6	Result Size: 303
[ 1 { 2    "name": "Four Horsemen Ale" 3 }, 4 { 5    "name": "INDIAne Pale Ale" 6 }, 7 { 8    "name": "Kolsch" 9 }, 10 { 11    "name": "Lake Effect Pale Ale" 12 }, 13 { 14    "name": "Raspberry Wheat Ale" 15 }, 16 { 17    "name": "Wall Street Wheat Ale" 18 ]				

```
SELECT * FROM `beer-sample` WHERE type="brewery" LIMIT 1;
```

The beer-sample bucket contains over 7000 documents, so the queries shown here contain a LIMIT clause to minimize the number of rows returned.

The following query returns all fields in one beer document. The IS NOT MISSING clause on the brewery\_id field tells N1QL to return only documents that have a brewery\_id field.

```
SELECT * FROM `beer-sample` WHERE brewery_id IS NOT MISSING  
AND type="beer" LIMIT 1;
```

The following query returns the brewery\_id and name fields from 5 beer documents:

```
SELECT brewery_id, name FROM `beer-sample`
  WHERE brewery_id IS NOT MISSING AND type="beer" LIMIT 5;
```

The screenshot shows the Couchbase Query Workbench interface. At the top, there is a toolbar with buttons for 'Execute' (highlighted in blue), 'Clear History', and 'Save Query'. Below the toolbar, the query is displayed in the editor:

```
SELECT brewery_id, name FROM `beer-sample`
  WHERE brewery_id IS NOT MISSING AND type="beer" LIMIT 5;
```

On the left side, there is a 'Bucket Analysis' panel listing various indexed fields and their types. Some fields are highlighted in green, such as 'name' and 'type'. The main area is titled 'Results' and shows the execution details: Status: success, Elapsed: 53.21ms, Execution: 53.16ms, Result Count: 5, and Result Size: 522. The results are presented as a JSON array of objects:

```

[{"brewery_id": "21st_amendment_brewery_cafe", "name": "21A IPA"}, {"brewery_id": "21st_amendment_brewery_cafe", "name": "563 Stout"}, {"brewery_id": "21st_amendment_brewery_cafe", "name": "Amendment Pale Ale"}, {"brewery_id": "21st_amendment_brewery_cafe", "name": "Bitter American"}]
```

You can view in the table format too:

The screenshot shows the Couchbase Query Workbench interface. At the top, there are three tabs: 'JSON' (selected), 'Table' (highlighted in blue), and 'Tree'. To the right of the tabs are buttons for 'Results' (large), 'Save JSON', and 'Cancel'. Below the tabs, status information is displayed: 'Status: success', 'Elapsed: 50.53ms', 'Execution: 50.49ms', 'Result Count: 5', and 'Result Size: 522'. The main area contains a table with two columns: 'brewery\_id' and 'name'. The data rows are: 21st\_amendment\_brewery\_cafe | 21A IPA, 21st\_amendment\_brewery\_cafe | 563 Stout, 21st\_amendment\_brewery\_cafe | Amendment Pale Ale, 21st\_amendment\_brewery\_cafe | Bitter American, and 21st\_amendment\_brewery\_cafe | Double Trouble IPA.

brewery_id	name
21st_amendment_brewery_cafe	21A IPA
21st_amendment_brewery_cafe	563 Stout
21st_amendment_brewery_cafe	Amendment Pale Ale
21st_amendment_brewery_cafe	Bitter American
21st_amendment_brewery_cafe	Double Trouble IPA

## Bucket Analysis

The bucket analysis area displays all the buckets installed in the cluster.

Bucket Analysis

Fully Queryable Buckets  
Queryable on Indexed Fields  
Non-Indexed Buckets  
▼ beer-sample

*Summary: 2 flavors found, sample size 1000 documents*

*Flavor 1 ( 19.9% ) , in-common: type = "brewery"*

- address (array)*
- city (string)*
- code (string)*
- country (string)*
- description (string)*
- geo (object), child type:*
  - accuracy (string)*
  - lat (number)*
  - lon (number)*
- name (string)*
- phone (string)*
- state (string)*
- type (string)*
- updated (string)*
- website (string)*

With the Enterprise Edition, you can expand the buckets to view the schema.  
Query Monitor

This catalog lists all currently executing active requests or queries.

```
SELECT * FROM system:active_requests;
```

The screenshot shows the Couchbase Query Workbench interface. At the top, there is a toolbar with buttons for 'Execute', 'Clear History', and 'Save Query'. Below the toolbar, a query editor window contains the SQL statement: 'SELECT \* FROM system:active\_requests;'. To the left of the main results area, there is a sidebar titled 'Bucket Analysis' which displays some summary statistics: 'Queryable Buckets', 'Queryable on Indexed Fields', '1-Indexed Buckets', and 'beer-sample'. A yellow summary message states: 'Summary: 2 flavors found, flavor 1 (19.9%), in-comm address (array) city (string)'. The main results area has tabs for 'JSON', 'Table' (which is selected), and 'Tree'. It shows the execution details: Status: success, Elapsed: 1.85ms, Execution: 1.78ms, Result Count: 1, and Result Size: 686. The results are presented in a table with the following columns: ClientContextID, ElapsedTime, ExecutionTime, PhaseCounts, and PhaseOperators. There is one row of data:

ClientContextID	ElapsedTime	ExecutionTime	PhaseCounts	PhaseOperators
0e8d46ca-7fb1-4e78-aa78-4c611909271b	1.455979ms	1.392735ms	PrimaryScan 2	Fetch 1 PrimaryScan 1

system:completed\_requests

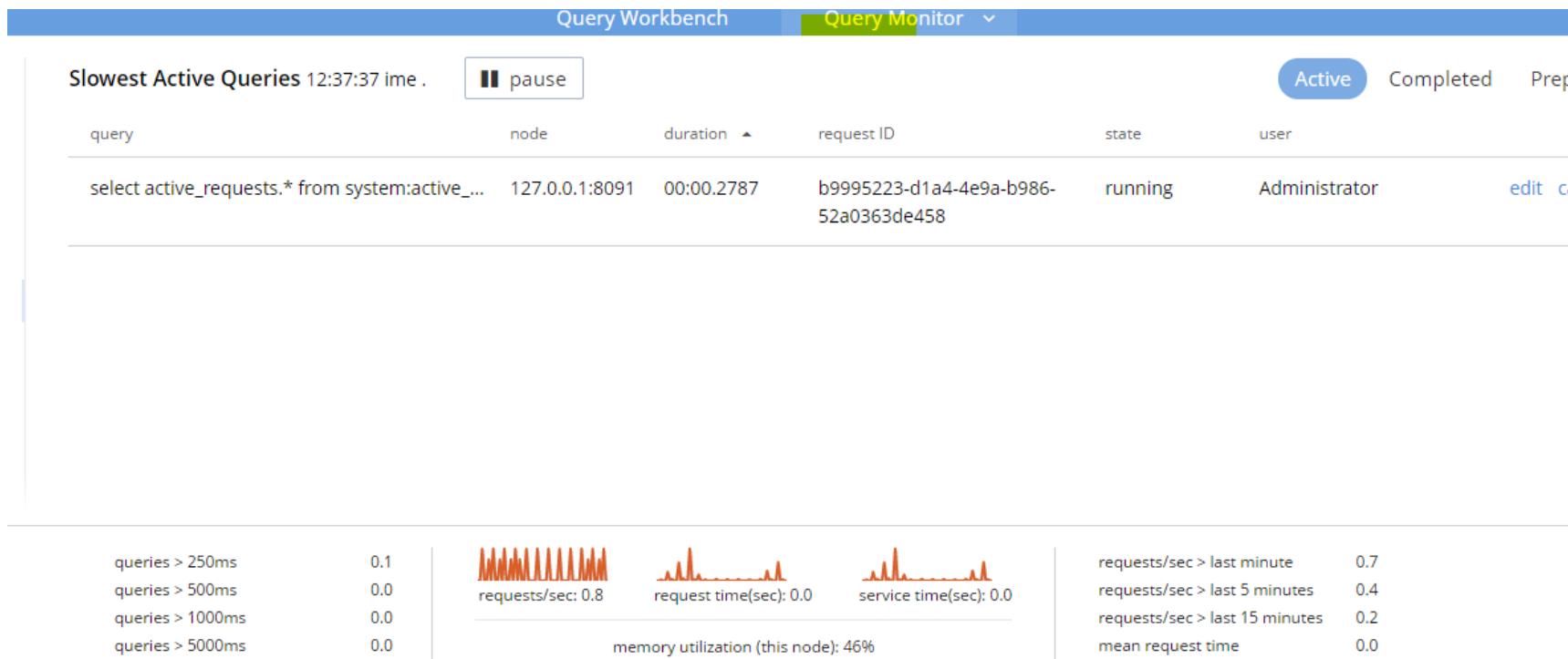
This catalog maintains a list of the most recent completed requests that have run longer than a predefined threshold of time. For each completed request, this catalog maintains information such as requestId, statement text, prepared name (if prepared statement), request time, service time, and so on. This information provides a general insight into the health and performance of the query engine and the cluster.

```
SELECT * FROM system:completed_requests LIMIT 1;
```

The screenshot shows the Couchbase Query Workbench interface. At the top, there are buttons for 'Execute' (highlighted in orange), 'Clear History', and 'Save Query'. Below the buttons is a search bar containing the query: 'SELECT \* FROM system:completed\_requests LIMIT 1;'. To the right of the search bar are navigation arrows and the text '15/15'. A 'Bucket Analysis' sidebar on the left provides summary information about queryable buckets and indexed fields. The main area is titled 'Results' and contains a table with the following data:

Status: success	Elapsed: 5.35ms	Execution: 5.30ms	Result Count: 1	Result Size: 584	
ClientContextID	Elapsed Time	ErrorCount	RequestId	ResultCount	ResultSize
78988fe3-b22c-4bf2-b2f7-071aad6a7885	5.218092366s	0	ce36d8d6-fe03-4282-b355-bf57b39599e1	0	0

You can even monitor the Query using the Query Monitor option.



Click on Competeted and Preparation tab to have a feel of the various option provided by the Monitoring console.

You should be able to create indexes using Query Work bench.

## Using the meta().id function

You can use the meta().id function when creating an index. The meta().id function does not require a parameter; it implicitly uses the keyspace being indexed.

All the Queries will be executed in the Query WB:

Query → Query Workbench

The screenshot shows the Couchbase Query Workbench interface. On the left, a sidebar lists navigation options: Dashboard, Servers, Buckets, Indexes, Search, **Query** (which is selected), Explain, XDCR, Security, Settings, and Logs. The main area has tabs for Query Editor, History (44/44), and Bucket Insights. The Bucket Insights panel shows 'Fully Queryable Buckets' containing 'beer-sample (7305)' and 'couchmusic2 (213063)', and 'Non-Indexed Buckets' containing 'default (1)'. Below these tabs is a 'Query Results' section with tabs for JSON, Table (which is selected), Tree, Plan, and Plan Text. A table displays the results of the query 'SELECT \* FROM system:active\_requests;'. The table has columns: clientContextID, elapsedTime, executionTime, node, phaseCounts, and phaseOperators. One row of data is shown:

clientContextID	elapsedTime	executionTime	node	phaseCounts	phaseOperators
ad79b98b-da62-4ca6-b557-8bd5985cd366	165.4852ms	165.4852ms	127.0.0.1:8091	primaryScan 1	authorize 1 fetch 1 primaryScan 1 au 1.00

`CREATE INDEX id_ix on `beer-sample`(meta().id);`

The `meta().id` function in a query is given an expression; if it resolves to a keyspace alias, the requested field (`id`) is returned.

```
SELECT b.name, meta(b).id
FROM `beer-sample` b
WHERE meta(b).id > "g" limit 1;
```

The screenshot shows the Couchbase Query Workbench interface. The top bar includes 'Execute', '19/19', 'Clear History', and 'Save Query'. The main area shows the query code:

```
SELECT b.name, meta(b).id
FROM `beer-sample` b
WHERE meta(b).id > "g" limit 1;
```

Below the code is a 'Bucket Analysis' section with tabs for Fully Queryable Buckets, Queryable on Indexed Fields, Non-Indexed Buckets, and a summary for 'beer-sample'. The 'Results' section shows the execution status: Status: success, Elapsed: 5.08ms, Execution: 5.04ms, Result Count: 1, Result Size: 93. The results table has columns: id and name. One row is shown:

id	name
g_heileman_brewing	G. Heileman Brewing

## Using indices for aggregates

If there is an index on the expression of an aggregate, that index may be used to satisfy the query. For example, given the index " abv\_idx" created using the following statement:

```
CREATE INDEX abv_idx ON `beer-sample`(abv);
```

The screenshot shows the Couchbase Query Workbench interface. The top navigation bar includes tabs for Overview, Server Nodes, Data Buckets, **Query**, Indexes, XDCR, Security, and Log. The Query tab is selected. Below the tabs is a toolbar with buttons for Execute, a history list (21/21), Clear History, and Save Query. The main area contains a code editor with the following SQL statement:

```
CREATE INDEX abv_idx ON `beer-sample`(abv);
```

On the left side, there is a sidebar titled "Bucket Analysis" with sections for Queryable Buckets, Queryable on Indexed Fields, and Indexed Buckets. The main results area is titled "Results" and shows the execution status: Status: success, Elapsed: 4.99s, Execution: 4.99s, Result Count: 0, Result Size: 0. Below this is a table titled "metrics" with columns: results, metrics, elapsedTime, executionTime, resultCount, and resultSize. The data row shows values: 4.986418541s, 4.98638084s, 0, 0.

The query engine will use the index " abv\_idx" for the following query:

```
SELECT min(abv), max(abv) FROM `beer-sample`;
```

The screenshot shows the Couchbase Query Workbench interface. The top navigation bar includes tabs for Overview, Server Nodes, Data Buckets, **Query**, Indexes, XDCR, Security, and Log. The Query tab is selected. Below the tabs is a toolbar with buttons for Execute, a history list (21/21), Clear History, and Save Query. The main area contains a code editor with the following SQL statement:

```
SELECT min(abv), max(abv) FROM `beer-sample`;
```

On the left side, there is a sidebar titled "Bucket Analysis" with sections for Queryable Buckets, Queryable on Indexed Fields, and Indexed Buckets. The main results area is titled "Results" and shows the execution status: Status: success, Elapsed: 948.83ms, Execution: 948.78ms, Result Count: 1, Result Size: 56. Below this is a table with columns \$1 and \$2. The data row shows values: 0 and 99.99.

The following example creates a secondary index that contains beers with an abv value greater than 5:  
**CREATE INDEX over5 ON `beer-sample`(abv) WHERE abv > 5 USING GSI;**

CREATE INDEX over5 ON `beer-sample`(abv) WHERE abv > 5 USING GSI;

results	metrics			
	elapsedTime	executionTime	resultCount	resultSize
[ ]	4.151164157s	4.151104109s	0	0

The following example creates a secondary index on the beer-sample bucket and then queries system:indexesfor status of the index:

```
CREATE INDEX `beer-sample-type-index` ON `beer-sample`(type) USING GSI;
```

```
SELECT * FROM system:indexes WHERE name="beer-sample-type-index";
```

CREATE INDEX `beer-sample-type-index` ON `beer-sample`(type) USING GSI;

results	metrics			
	elapsedTime	executionTime	resultCount	resultSize
[ ]	5.488963818s	5.488885563s	0	0

The screenshot shows the Couchbase Query Workbench interface. At the top, there are buttons for 'Execute', 'Clear History', and 'Save Query'. Below that is a code editor containing the SQL-like query: `SELECT * FROM system:indexes WHERE name="beer-sample-type-index";`. The results are displayed in a table titled 'Results' with columns: `datastore_id`, `id`, `index`, `key`, `keyspace_id`, `name`, and `namespace`. The data shown is:

<code>datastore_id</code>	<code>id</code>	<code>index</code>	<code>key</code>	<code>keyspace_id</code>	<code>name</code>	<code>namespace</code>
http://127.0.0.1:8091	8344059b3d68664e	`type`	beer-sample	beer-sample-type-index	beer-sample-type-index	default

## Limitations

The total size of the index keys cannot exceed 4K for a single document. Index key size is calculated using the total size of all the expressions being indexed in a single document. If an index keys size exceeds 4K, it will be skipped. The following error is logged to indicate that an item is skipped when building the index: "Encoded secondary key is too long" in the indexer.log file. The indexer.log file is included in cbcollect\_info output.

You can view all the indexes as shown below: Indexes → Global Indexes

The screenshot shows the Couchbase Admin UI with the 'Indexes' tab selected. On the left, there is a sidebar with navigation links: Dashboard, Servers, Buckets, **Indexes**, Search, Query, XDCR, Security, Settings, and Logs. The main area displays a table titled 'Global Indexes' with the following columns: bucket ▾, node, index name, storage type, status, and build progress. The data in the table is as follows:

bucket ▾	node	index name	storage type	status	build progress
beer-sample	127.0.0.1:8091	abv_idx	Standard GSI	ready	100%
beer-sample	127.0.0.1:8091	beer_primary	Standard GSI	ready	100%
beer-sample	127.0.0.1:8091	id_ix	Standard GSI	ready	100%
couchmusic2	127.0.0.1:8091	#primary	Standard GSI	ready	100%
couchmusic2	127.0.0.1:8091	idx_country_population	Standard GSI	ready	100%
couchmusic2	127.0.0.1:8091	idx_email	Standard GSI	ready	100%
couchmusic2	127.0.0.1:8091	idx_postalCode	Standard GSI	ready	100%

-----End of Lab-----

### 13. N1QL : Load data in Bucket, create primary index – 35 Minutes

#### Objectives

A	Install and configure Couchbase Server .x as a single node cluster
B	Survey the Couchbase administration console
C	Create and configure a Couchbase bucket
D	Load documents into Couchbase using <i>cbimport</i> or <i>cbdocloader</i>
E	Create a primary index to support ad hoc queries (CREATE PRIMARY INDEX)

If you have an existing Couchbase installation you wish to restore later, archive the data and configuration files from these locations.

#### Create and configure a Couchbase bucket

*You want to create the basic Couchbase data container, a bucket.*

In the Couchbase UI, select the *Buckets* panel, and choose *Add Bucket*.

The screenshot shows the Couchbase Admin UI for a 'Training Cluster'. The top navigation bar includes links for Activity, Classic UI, Documentation, Support, and Administrator. On the left, a sidebar menu lists Dashboard, Servers, Buckets (which is highlighted with a yellow starburst icon), Indexes, and Search. The main content area is titled 'Training Cluster > Buckets' and displays a message: 'You have no data buckets.' Below this, there are sections for 'name' and 'items' (both currently empty) and 'disk used' (also empty). A red arrow points from the 'Buckets' menu item in the sidebar to the 'You have no data buckets.' message. Another red arrow points from the 'ADD BUCKET' button in the top right to the 'disk used' section.

7. In the *Add Data Bucket* panel, set the following values.

- Name: *couchmusic2*
- Memory Quota: *1024mb*
- Advanced Settings (*click*)
  - Replicas: *Disable (uncheck)*
  - Flush: *Enable (check)*

Note, because replicas are created on distinct nodes, replication requires a minimum of two nodes. So, for this single node training cluster, you will disable replication for this bucket. In production, Couchbase recommends a minimum of three nodes per cluster.

Add Data Bucket X

Name

Memory Quota in megabytes per server node  
1024 MB  
other buckets (0 B) this bucket (1 GB) remaining (1 GB)

Bucket Type  
 Couchbase  Memcached  Ephemeral

 Advanced bucket settings

Replicas  
 Enable  Replicate view indexes

Conflict Resolution ⓘ  
 Sequence number  Timestamp

Ejection Method ⓘ  
 Value-only  Full

Bucket Priority ⓘ  
 Default  High

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

Flush ⓘ  
  Enable

[Cancel](#) [Add Bucket](#)

8. Click the bucket name to review summary information for the new *couchmusic2* bucket.

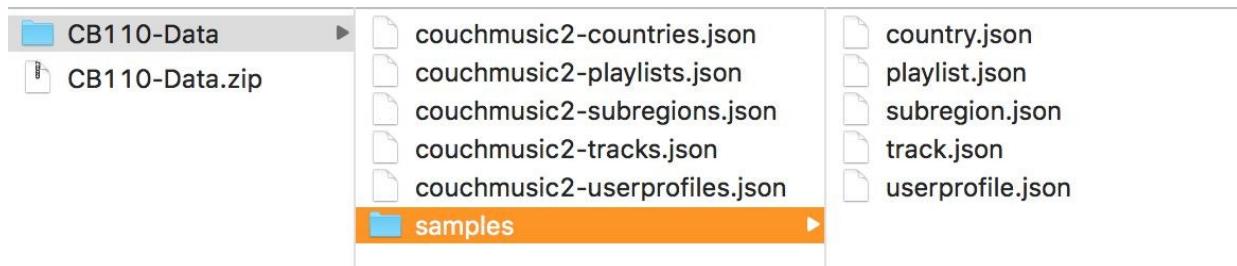
Training Cluster > Buckets ADD BUCKET

name ▾	items	resident	ops/sec	RAM used/quota	disk used	
couchmusic2 	0	100%	0	1.83MB / 1GB	269KB	<a href="#">Documents</a> <a href="#">Statistics</a>
<p>Type: Couchbase Bucket RAM Quota: 1GB Cluster RAM Quota: 2GB Replicas: disabled Server Nodes: 1 Ejection Method: Value-Only Conflict Resolution: Sequence Number Compaction: Not active</p> <p>Memory</p> <p>Disk</p> <p>total cluster storage (464 GB)</p> <p>other buckets (0 B) this bucket (269 KB) remaining (41.8 GB)</p>						<span style="border: 1px solid #ccc; padding: 2px 10px;">Delete</span> <span style="border: 1px solid #ccc; padding: 2px 10px;">Compact</span> <span style="border: 1px solid #ccc; padding: 2px 10px;">Flush</span> <span style="background-color: #0072bc; color: white; border: 1px solid #0072bc; padding: 2px 10px;">Edit</span>

## A. Load documents into Couchbase using cbimport

You want to bulk load data into the bucket you have created.

9. Use the *CB110-Data.zip* file and extract it to your *desktop* or similar directory. It contains JSON files of five different types of JSON document (object), related to an application named *couchmusic2*, along with a PDF document describing their structure.



10. Open a Terminal (Command) window, navigate to the Couchbase *bin* folder, and briefly review the contents of this folder. Notice the *cbimport*, *cbq*, and *couchbase-cli* tools.

macOS	/Applications/Couchbase Server.app/Contents/Resources/couchbase-core/bin/
Windows	C:\Program Files\Couchbase\Server\bin\
Linux	/opt/couchbase/bin/

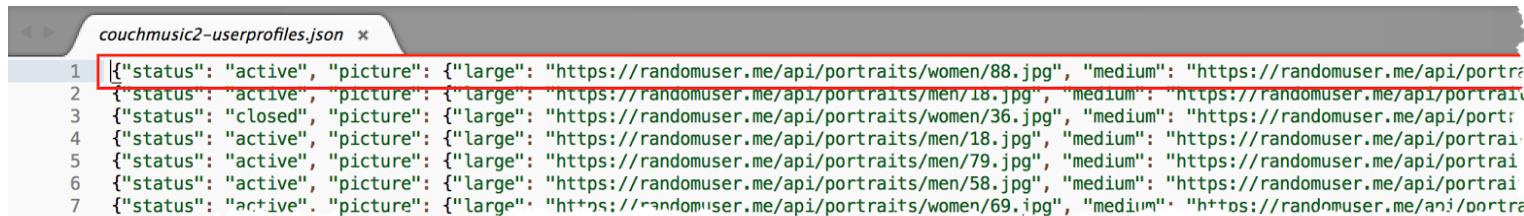
```
[couchbase:bin Schuman$ ls
c_rehash
cbbackup
cbbackupmgr
cbbackupwrapper
cbbrowse_logs
cbccollect_info
cbcompact
cbdocloader
cbdumpp-config
cbenable_core_dumps.sh
cbeectl
cbexport
cbft
cbft-bleve
cbimport
cbindex
cbindexperf
cbindexplan
cbq
cbq-engine
cbq.old
cbrecovery
cbrestore
cbvbucketctl
cbvdiff
cbworkloadgen
couch_compact
couch_dbck
couch_dbdump
couch_dbinfo
couch_view_file_merger
couch_view_group_cleanup
couch_view_group_compactor
couch_view_index_builder
couch_view_index_updater
couchbase-cli
couchbase-server
couchdb
couchjs
couchjs.tpl
ct_run
dbdiff
dump-guts
dump-stats
epmd
erl
gometa
goport
go secrets
go xdcr
go zip
indexer
install
jeprof
kv_trace_dump
mc_bp_packet_printer
mc_ctl
mc logsplit
mc stat
mc timings
memcached
mossScope
moxi
openssl
plasma_dump
priv
projector
saslauthd_port
sigar_port]
```

Note, *cbimport* is a robust data loading tool released as a new feature of Couchbase 5.0. For full detail on using *cbimport*, see here:

<https://developer.couchbase.com/documentation/server/current/tools/cbimport.html>

11. Add this *bin* folder to the PATH environment variable for your operating system, so that its commands may be invoked from any command line location. If needed, look up your operating system documentation for details on this process.

12. In the Terminal, navigate back to the */CB110-Data* folder.
13. In a text editor, open *CB110-Data/couchmusic2-countries.json*, and examine its contents. Notice each line contains a JSON document (object).



```

couchmusic2-userprofiles.json *
1 [{"status": "active", "picture": {"large": "https://randomuser.me/api/portraits/women/88.jpg", "medium": "https://randomuser.me/api/portrait/women/88.jpg", "small": "https://randomuser.me/api/portraits/thumb/women/88.jpg"}, "age": 32, "name": {"first": "Aahinge", "last": "Fetteness"}, "location": {"city": "Bogot\u00e1", "country": "Colombia"}, "gender": "female", "type": "userprofile", "username": "aahingeffeteness42037", "email": "delores.riley@hotmail.com", "title": "Mrs", "favoriteGenres": ["Classical Crossover", "Contemporary Blues", "French Pop", "Progressive Bluegrass"], "dateOfBirth": "1983-07-12", "pwd": "636172626f6e"}, {"status": "active", "picture": {"large": "https://randomuser.me/api/portraits/men/18.jpg", "medium": "https://randomuser.me/api/portrait/men/18.jpg", "small": "https://randomuser.me/api/portraits/thumb/men/18.jpg"}, "age": 28, "name": {"first": "Delores", "last": "Riley"}, "location": {"city": "Bogot\u00e1", "country": "Colombia"}, "gender": "female", "type": "userprofile", "username": "delores.riley", "email": "delores.riley@hotmail.com", "title": "Mrs", "favoriteGenres": ["Classical Crossover", "Contemporary Blues", "French Pop", "Progressive Bluegrass"], "dateOfBirth": "1990-05-12", "pwd": "636172626f6e"}, {"status": "closed", "picture": {"large": "https://randomuser.me/api/portraits/women/36.jpg", "medium": "https://randomuser.me/api/portrait/women/36.jpg", "small": "https://randomuser.me/api/portraits/thumb/women/36.jpg"}, "age": 36, "name": {"first": "Aahinge", "last": "Fetteness"}, "location": {"city": "Bogot\u00e1", "country": "Colombia"}, "gender": "female", "type": "userprofile", "username": "aahingeffeteness42037", "email": "delores.riley@hotmail.com", "title": "Mrs", "favoriteGenres": ["Classical Crossover", "Contemporary Blues", "French Pop", "Progressive Bluegrass"], "dateOfBirth": "1983-07-12", "pwd": "636172626f6e"}, {"status": "active", "picture": {"large": "https://randomuser.me/api/portraits/men/18.jpg", "medium": "https://randomuser.me/api/portrait/men/18.jpg", "small": "https://randomuser.me/api/portraits/thumb/men/18.jpg"}, "age": 28, "name": {"first": "Delores", "last": "Riley"}, "location": {"city": "Bogot\u00e1", "country": "Colombia"}, "gender": "female", "type": "userprofile", "username": "delores.riley", "email": "delores.riley@hotmail.com", "title": "Mrs", "favoriteGenres": ["Classical Crossover", "Contemporary Blues", "French Pop", "Progressive Bluegrass"], "dateOfBirth": "1990-05-12", "pwd": "636172626f6e"}, {"status": "active", "picture": {"large": "https://randomuser.me/api/portraits/men/79.jpg", "medium": "https://randomuser.me/api/portrait/men/79.jpg", "small": "https://randomuser.me/api/portraits/thumb/men/79.jpg"}, "age": 39, "name": {"first": "Aahinge", "last": "Fetteness"}, "location": {"city": "Bogot\u00e1", "country": "Colombia"}, "gender": "female", "type": "userprofile", "username": "aahingeffeteness42037", "email": "delores.riley@hotmail.com", "title": "Mrs", "favoriteGenres": ["Classical Crossover", "Contemporary Blues", "French Pop", "Progressive Bluegrass"], "dateOfBirth": "1983-07-12", "pwd": "636172626f6e"}, {"status": "active", "picture": {"large": "https://randomuser.me/api/portraits/men/58.jpg", "medium": "https://randomuser.me/api/portrait/men/58.jpg", "small": "https://randomuser.me/api/portraits/thumb/men/58.jpg"}, "age": 30, "name": {"first": "Delores", "last": "Riley"}, "location": {"city": "Bogot\u00e1", "country": "Colombia"}, "gender": "female", "type": "userprofile", "username": "delores.riley", "email": "delores.riley@hotmail.com", "title": "Mrs", "favoriteGenres": ["Classical Crossover", "Contemporary Blues", "French Pop", "Progressive Bluegrass"], "dateOfBirth": "1990-05-12", "pwd": "636172626f6e"}, {"status": "active", "picture": {"large": "https://randomuser.me/api/portraits/women/69.jpg", "medium": "https://randomuser.me/api/portrait/women/69.jpg", "small": "https://randomuser.me/api/portraits/thumb/women/69.jpg"}, "age": 30, "name": {"first": "Aahinge", "last": "Fetteness"}, "location": {"city": "Bogot\u00e1", "country": "Colombia"}, "gender": "female", "type": "userprofile", "username": "aahingeffeteness42037", "email": "delores.riley@hotmail.com", "title": "Mrs", "favoriteGenres": ["Classical Crossover", "Contemporary Blues", "French Pop", "Progressive Bluegrass"], "dateOfBirth": "1990-05-12", "pwd": "636172626f6e"}]

```

14. Open the sample of this document type, *CB110-Data/samples/userprofile.json*, and examine its structure. Notice its *type* and *username* properties.



```

userprofile.json *
1 {
2   "status": "active",
3   "picture": {
4     "large": "https://randomuser.me/api/portraits/women/88.jpg",
5     "medium": "https://randomuser.me/api/portrait/women/88.jpg",
6     "small": "https://randomuser.me/api/portraits/thumb/women/88.jpg"
7   },
8   "age": 32,
9   "name": {
10     "first": "Aahinge",
11     "last": "Fetteness"
12   },
13   "location": {
14     "city": "Bogot\u00e1",
15     "country": "Colombia"
16   },
17   "gender": "female",
18   "type": "userprofile",
19   "username": "aahingeffeteness42037",
20   "email": "delores.riley@hotmail.com",
21   "title": "Mrs",
22   "favoriteGenres": [
23     "Classical Crossover",
24     "Contemporary Blues",
25     "French Pop",
26     "Progressive Bluegrass"
27   ],
28   "dateOfBirth": "1983-07-12",
29   "pwd": "636172626f6e"
30 }

```

15. Use *cbimport* to load *country* JSON documents to the *couchmusic2* bucket. Use a key pattern to generate a key for each document, as shown below.

- ❑ Cluster (-c): *couchbase://127.0.0.1*
- ❑ Username (-u): *Administrator*
- ❑ Password (-p): *password*
- ❑ Bucket (-b): *couchmusic2*
- ❑ Format (-f): *lines*
- ❑ Dataset (-d): *file://couchmusic2-userprofiles.json*
- ❑ Threads (-t): *2*
- ❑ Key Pattern to Generate (-g): *%type%::%username%*

```
cbimport json -c couchbase://127.0.0.1 -u Administrator -p password -b couchmusic2 -f lines  
-d file://couchmusic2-userprofiles.json -t 2 -g %type%::%username%
```

```
D:\MyDocuments\Workspace\CB\Labs\CB110-Data>cbimport json -c couchbase://192.168.139.129 -u Administrator -p admin123 -b couchmusic2 -f lines -d file://couchmusic2-userprofiles.json -t 2 -g %type%::%username%  
Json `file://couchmusic2-userprofiles.json` imported to `http://192.168.139.129:8091` successfully
```

**Note, if you are using Windows 10,** you must escape % delimiters on the command line, using the ^ character, if the variable name referenced using those delimiters is also used as a system variable name, such as *username*. Modify the command show as:

```
cbimport json -c couchbase://127.0.0.1 -u Administrator -p password -b couchmusic2 -f lines  
-d file://couchmusic2-userprofiles.json -t 2 -g %type%::^%username^%
```

or

```
cbimport json -c couchbase://192.168.139.129 -u Administrator -p admin123 -b couchmusic2 -f lines -d  
file://couchmusic2-userprofiles.json -t 2 -g %type%::^%username^%
```

```
D:\MyDocuments\Workspace\CB\Labs\CB110-Data>cbimport json -c couchbase://192.168.139.129 -u Administrator -p admin123 -b  
couchmusic2 -f lines -d file://couchmusic2-userprofiles.json -t 2 -g %type%::^%username^%
```

```
Json `file://couchmusic2-userprofiles.json` imported to `http://192.168.139.129:8091` successfully
```

```
D:\MyDocuments\Workspace\CB\Labs\CB110-Data>
```

```
[couchbase:CB110-Data Schuman$ ls
couchmusic2-countries.json      couchmusic2-subregions.json      samples
couchmusic2-data-model.pdf       couchmusic2-tracks.json
couchmusic2-playlists.json       couchmusic2-userprofiles.json
[couchbase:CB110-Data Schuman$ cbimport json -c couchbase://127.0.0.1 -u Administrator -p password ]
-b couchmusic2 -f lines -d file://couchmusic2-userprofiles.json -t 2 -g %type%::%username%
Json `file://couchmusic2-userprofiles.json` imported to `http://127.0.0.1:8091` successfully
couchbase:CB110-Data Schuman$ ]
```

16. In the Couchbase UI, open the *couchmusic2* bucket, examine the number of documents loaded, and open the Documents screen.

name ▾	items	resident	ops/sec	RAM used/quota	disk used
couchmusic2	49,981	100%	0	52.3MB / 1GB	37.7MB

17. In the Documents screen, open the first document for editing.

Training Cluster > Buckets > Documents

ADD DOCUMENT

Dashboard    couchmusic2    filter: ?skip=0&include\_docs=true&limit=11

ID    content sample

userprofile::aahingeffecteness42037    {"status":"active","picture":{"large":"https://randomuser.me/api/portraits/women/88.jpg","medium":"https://randomuser.me/api/portraits/med/women/88.jpg","thumbnail":"https://randomuser.me/api/portrait"}  
Delete    Edit

userprofile::aahingheadwaiter24314    {"status":"active","picture":{"large":"https://randomuser.me/api/portraits/men/18.jpg","medium":"https://randomuser.me/api/portraits/med/men/18.jpg","thumbnail":"https://randomuser.me/api/portrait"}  
Delete    Edit

18. Notice the metadata related to this document.

Training Cluster > Documents > Documents Editing

userprofile::aahingeffecteness42037

Dashboard    userprofile::aahingeffecteness42037    Delete    Save As...    Save

```

1 {
2   "status": "active",
3   "picture": {
4     "large": "https://randomuser.me/api/portraits/women/88.jpg",
5     "medium": "https://randomuser.me/api/portraits/med/women/88.jpg",
6     "thumbnail": "https://randomuser.me/api/portraits/thumb/women/88.jpg"
7   },
8   "gender": "female",
9   "firstName": "Delores",
10  "created": "2015-01-18T10:58:26",
11  "phones": [
12    {
13      "type": "home",
14      "verified": "2015-09-16T07:02:39",
15      "number": "(943)-434-3888"
16    }
17  ],
18  "addresses": []

```

19. Modify and run *cbimport* four more times as shown below, to load each of the remaining document sets, using their type and the specified value from each

document as a key.

```
cbimport json -c couchbase://127.0.0.1 -u Administrator -p password -b couchmusic2 -f  
lines -d file:///couchmusic2-playlists.json -t 2 -g %type%::%id%
```

```
cbimport json -c couchbase://127.0.0.1 -u Administrator -p password -b couchmusic2 -f  
lines -d file://couchmusic2-subregions.json -t 2 -g %type%::%region-number%
```

```
cbimport json -c couchbase://127.0.0.1 -u Administrator -p password -b couchmusic2 -f  
lines -d file://couchmusic2-tracks.json -t 2 -g %type%::%id%
```

```
cbimport json -c couchbase://127.0.0.1 -u Administrator -p password -b couchmusic2 -f  
lines -d file://couchmusic2-countries.json -t 2 -g %type%::%countryCode%
```

Note, it is a common JSON design pattern to prefix document keys with a type identifier (e.g., *country*::*ES*). When using *cbimport*, prefixing can be specified using key generation, as shown above.

Windows:

```
cbimport json -c couchbase://192.168.139.129 -u Administrator -p admin123 -b couchmusic2 -f  
lines -d file://couchmusic2-playlists.json -t 2 -g %type%::^%id^%
```

```
cbimport json -c couchbase://192.168.139.129 -u Administrator -p admin123 -b couchmusic2 -f  
lines -d file://couchmusic2-subregions.json -t 2 -g %type%::^%region-number^%
```

```
cbimport json -c couchbase://192.168.139.129 -u Administrator -p admin123 -b couchmusic2 -f  
lines -d file://couchmusic2-tracks.json -t 2 -g %type%::^%id^%
```

```
cbimport json -c couchbase://192.168.139.129 -u Administrator -p admin123 -b couchmusic2 -f
```

```
lines -d file://couchmusic2-countries.json -t 2 -g %type%::^%countryCode^%
```

```
D:\MyDocuments\Workspace\CB\Labs\CB110-Data>cbimport json -c couchbase://192.168.139.129 -u Administrator -p admin123 -b couchmusic2 -f lines -d file://couchmusic2-subregions.json -t 2 -g %type%::^%region-number^%
Json `file://couchmusic2-subregions.json` imported to `http://192.168.139.129:8091` successfully

D:\MyDocuments\Workspace\CB\Labs\CB110-Data>cbimport json -c couchbase://192.168.139.129 -u Administrator -p admin123 -b couchmusic2 -f lines -d file://couchmusic2-tracks.json -t 2 -g %type%::^%id^%
Json `file://couchmusic2-tracks.json` imported to `http://192.168.139.129:8091` successfully

D:\MyDocuments\Workspace\CB\Labs\CB110-Data>cbimport json -c couchbase://192.168.139.129 -u Administrator -p admin123 -b couchmusic2 -f lines -d file://couchmusic2-countries.json -t 2 -g %type%::^%countryCode^%
Json `file://couchmusic2-countries.json` imported to `http://192.168.139.129:8091` successfully

D:\MyDocuments\Workspace\CB\Labs\CB110-Data>cbimport json -c couchbase://192.168.139.129 -u Administrator -p admin123 -b couchmusic2 -f lines -d file://couchmusic2-playlists.json -t 2 -g %type%::^%id^%
Json `file://couchmusic2-playlists.json` imported to `http://192.168.139.129:8091` successfully

D:\MyDocuments\Workspace\CB\Labs\CB110-Data>
```

20. In the Couchbase UI, verify you've loaded 213,063 documents to a *couchmusic2* bucket, and open the *Documents* screen.

name ▾	items	resident	ops/sec	RAM used/quota	disk used	Documents	Statistics
couchmusic2	213,063	100%	0	247MB / 1GB	173MB	<a href="#">Documents</a>	<a href="#">Statistics</a>

Type: Couchbase  
Bucket RAM Quota: 1GB  
Cluster RAM Quota: 1.99GB  
Replicas: disabled  
Server Nodes: 1  
Ejection Method: Value-Only  
Conflict Resolution: Sequence Number  
Compaction: Not active

Memory

Disk

cluster quota (1.99 GB)

total cluster storage (464 GB)

other buckets (0 B)  
this bucket (1 GB)  
remaining (1023 MB)

other buckets (0 B)  
this bucket (173 MB)  
remaining (32.5 GB)

Delete Compact Flush Edit

## Create a primary index to support ad hoc queries (CREATE PRIMARY INDEX)

*You want to enable ad hoc N1QL querying for the bucket you've created.*

19. In the Couchbase UI, select the Query tab to open the Query Workbench.
20. To allow ad hoc N1QL queries to be run on the *couchmusic2* bucket, run a *CREATE PRIMARY INDEX* statement on this bucket, using the global indexing service.

```
CREATE PRIMARY INDEX ON `couchmusic2` USING GSI;
```

The screenshot shows the Couchbase Query Workbench. On the left, a sidebar lists navigation options: Dashboard, Servers, Buckets, Indexes, Search, Query (highlighted with a yellow starburst), XDCR, and Security. The main area is divided into two sections: 'Query Editor' and 'Bucket Insights'. In the 'Query Editor', a single query is listed: 'CREATE PRIMARY INDEX ON `Customer360` USING GSI;'. Below the query are buttons for 'Execute' (highlighted with a yellow starburst) and 'Explain'. A status message indicates 'success | elapsed: 229.11ms | execution: 229.09ms | count: 0 | size: 0'. To the right, the 'Bucket Insights' panel displays information for the 'Customer360' bucket, specifically noting it is 'Fully Queryable' and 'Queryable on Indexed Fields'. Below the 'Bucket Insights' panel are sections for 'Non-Indexed Buckets'.

Note, once a primary index is created on a bucket, it will appear as "Fully Queryable" in the *Bucket Insights* display. A primary index enables ad hoc queries for development and learning purposes.

On production systems, you may choose to drop the primary index to improve system performance by eliminating the cost of maintaining it for new writes.

```
DROP PRIMARY INDEX ON [bucket name] USING GSI;
```

**Do not drop the primary index** on *couchmusic2*, as it will be relied upon in later Labs in this course, for learning purposes.

-----End of Lab-----

**14. N1QL - Select documents in *Workbench* and the *cbq* tool – 60 Minutes**

## Objectives

A	Use Query Workbench to select all attributes from all documents, save the results, and limit selection (*, SELECT, LIMIT)
B	Cancel a long-running query
C	(Optional) Execute a query using the <i>cbq</i> command line query tool

A. *Use Query Workbench to select all attributes from all documents, and limit selection (\*, SELECT, LIMIT)*

1. In the Couchbase Server UI, select the *Query* tab to open the *Query Workbench*.
2. Write and execute a query to SELECT all attributes from all documents FROM *couchmusic2*, but LIMIT your result to 10 documents.

```
SELECT *  
FROM couchmusic2 LIMIT 10;
```

Notice the *Status*, *Elapsed*, *Execution*, *Result Count*, and *Result Size* metrics displayed over the result, and review the other display and storage capabilities of the UI.

The screenshot shows the Couchbase Query Workbench interface. At the top, there are tabs for "Query Workbench" and "Query Monitor". Below the tabs, the "Query Editor" contains the following SQL-like query:

```
1 SELECT *
2 FROM couchmusic2
3 LIMIT 10;
```

The "Execute" button is highlighted with a red box. To its right are "Explain" and "success | elapsed: 10.83ms | execution: 10.81ms | count: 10 | size: 3177". Further right are "Preferences" and a "history (2/2)" button with navigation arrows, also highlighted with a red box.

The "Query Results" section displays the JSON output of the query. The results are paginated, with the first page shown. The "JSON" tab is selected and highlighted with a red box. Other tabs include "Table", "Tree", "Plan", and "Plan Text".

```
1 [ ]
2 {
3   "couchmusic2": {
4     "countryCode": "AD",
5     "gdp": 40214,
6     "name": "Andorra",
7     "population": 80792,
8     "region-number": 39,
9     "type": "country",
10    "updated": "2015-10-01T07:35:13"
11  }
12 },
13 { }
```

### 3. Toggle between different displays for the results.

Query Results

countryCode	gdp	name	population	region-number	type	updated
AD	40214	Andorra	80792	39	country	2015-10-01T07:35:13
AE	41433	United Arab Emirates	9693829	145	country	2015-09-11T08:31:38
AF	650	Afghanistan	32059823	34	country	2015-09-03T21:08:49
AG	13812	Antigua and Barbuda	91873	29	country	2015-09-12T19:47:41
AI	0	Anguilla	14640	29	country	2015-09-04T22:15:09
AL	4184	Albania	3195196	39	country	2015-09-12T20:19:13
AM	3640	Armenia	2989427	145	country	2015-09-01T20:26:21
AN	0	Netherlands Antilles	0	29	country	2015-09-27T05:45:24
AO	5750	Angola	22850336	17	country	2015-09-16T13:43:28
AP	0	Asia Pacific	0	990	country	2015-09-26T02:02:08

### 4. Open the query export tool.

The screenshot shows the Couchbase Query Workbench interface. At the top, there's a navigation bar with a logo, the text "Training Cluster > Query", and a dropdown menu "Query Workbench". To the right of the dropdown is a red arrow pointing to the "EXPORT" button, which is located on a blue ribbon-like bar. Below the navigation bar, there's a sidebar with links for "Dashboard", "Servers", "Buckets", and "Query Editor". The "Query Editor" section contains a code editor with the following SQL query:

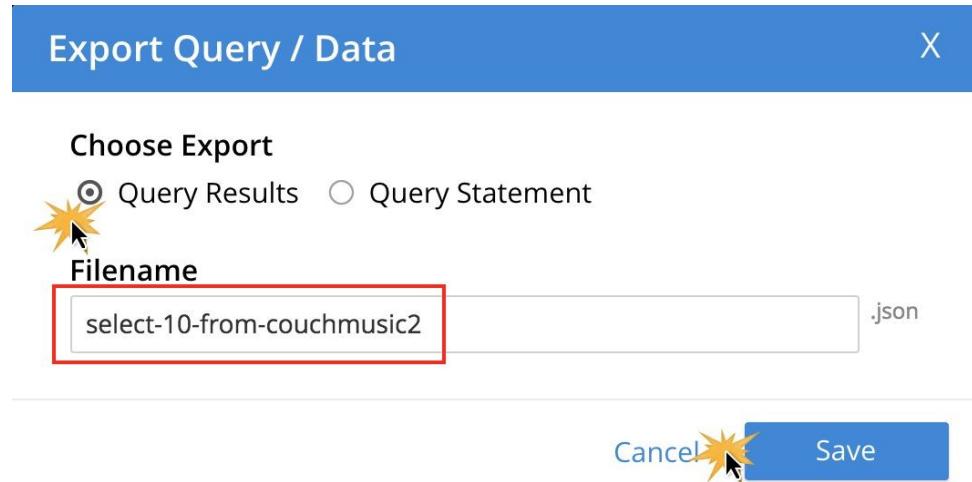
```

1 SELECT *
2 FROM couchmusic2
3 LIMIT 10;

```

On the right side of the interface, there are sections for "Primary Buckets" (listing "couchbase" and "couchbase\_system") and "Replica Buckets" (listing "couchbase\_repl" and "couchbase\_system\_repl").

### 5. Export the query results to the desktop as a JSON file.



6. Examine the query results in a text editor.

```
▶ select-10-from-couchmusic2.json ×  
1 [  
2 {  
3   "couchmusic2": {  
4     "countryCode": "AD",  
5     "gdp": 40214,  
6     "name": "Andorra",  
7     "population": 80792,  
8     "region-number": 39,  
9     "type": "country",  
10    "updated": "2015-10-01T07:35:13"  
11  }  
12 },  
13 {  
14   "couchmusic2": {
```

## Cancel a long-running query

7. Modify the prior query by removing the LIMIT clause, then execute the query.

```
SELECT *
FROM couchmusic2;
```

Notice the *Execute* button changes to a *Cancel* button during long query execution.

The screenshot shows the Couchbase Query Editor interface. At the top, there's a toolbar with a 'history (3/3)' button. Below it is a code editor window containing the following SQL query:

```
1 SELECT *
2 FROM couchmusic2;
```

Below the code editor, there's a status bar with buttons for 'Cancel' (which is highlighted with a red box), 'Explain', and various metrics: 'Executing | elapsed: | execution: | count: | size: 0'. To the right of the status bar is a 'Preferences' button. Further down, there's a 'Query Results' section with a table header row and a single data row:

1 {"status": "Executing statement"}
1 {"status": "Executing statement"}

At the bottom of the results table, there are tabs for 'JSON', 'Table', 'Tree', 'Plan', and 'Plan Text'. The 'JSON' tab is currently selected.

8. Cancel the running query.

B. (Optional) Execute a query using the cbq command line query tool

9. In a Terminal window, launch the *cbq* tool located in the Couchbase *bin* directory.

Authenticate to your local database engine using the Administrator credentials you created above.

```
cbq -e 192.168.139.129:8091 -u Administrator -p admin123
```

```
[couchbase:CB110-Data Schuman$  
[couchbase:CB110-Data Schuman] cbq -e localhost:8091 -u Administrator -p password  
Connected to : http://localhost:8091/. Type Ctrl-D or \QUIT to exit.  
Path to history file for the shell : /Users/Schuman/.cbq_history  
cbq> 
```

10. Select all attributes from *couchmusic2*, limiting the results to 1 document, and compare the resulting output to that seen when using Query Workbench.

```
SELECT *  
FROM couchmusic2 LIMIT 1;
```

```
[couchbase:CB110-Data Schuman$  
[couchbase:CB110-Data Schuman$ cbq -e localhost:8091 -u Administrator -p password  
  Connected to : http://localhost:8091/. Type Ctrl-D or \QUIT to exit.  
  
  Path to history file for the shell : /Users/Schuman/.cbq_history  
cbq> SELECT *  
  > FROM couchmusic2  
  > LIMIT 1;  
{  
  "requestID": "1cd43c73-2256-4ed1-88b3-2cd1d0378157",  
  "signature": {  
    "*": "*"  
  },  
  → "results": [  
    {  
      "couchmusic2": {  
        "countryCode": "AD",  
        "gdp": 40214,  
        "name": "Andorra",  
        "population": 80792,  
        "region-number": 39,  
        "type": "country",  
        "updated": "2015-10-01T07:35:13"  
      }  
    }  
  ],  
  → "status": "success",  
  → "metrics": {  
    "elapsedTime": "6.6608ms",  
    "executionTime": "6.64484ms",  
    "resultCount": 1,  
    "resultSize": 314  
  }  
}  
cbq> █
```

10. Examine the help for cbq.

```
[couchbase:CB110-Data Schuman$ [couchbase:CB110-Data Schuman$ cbq -h
Usage of cbq:
-b string
    Shorthand for -batch (default "off")
-batch string
    Batch mode for sending queries to Asterix. Values : on/off (default "off")
-c string
    Shorthand for -credentials
-credentials string
    A list of credentials, in the form user:password.
    For example : -c beer-sample:pass
-e string
    Shorthand for -engine (default "http://localhost:8091/")
-engine string
    URL to the query service/cluster.
    Default : http://localhost:8091
```

## Indexes: Specified document attributes and selecting by index values

### *Objectives*

A	Filter queries by attribute values (WHERE)
A	Implement an index for a specific document attribute (CREATE INDEX)
B	Create and use an index for a multiply filtered query (AND)
C	Implement an index for an attribute and filter (CREATE

	INDEX ... WHERE)
--	------------------

Note, this lab assumes you are already familiar with using the *AND* keyword in SQL, when applying multiple filter clauses to a query.

Filter queries by attribute values (WHERE)

1. Select the *address* of a document with the *email* value "delores.riley@hotmail.com".

```
SELECT address FROM  
couchmusic2  
WHERE email = "delores.riley@hotmail.com";
```

Notice the relatively long execution time for this query, due to the full bucket scan it requires. The specific time will vary by system, but will be longer than necessary.

Query Editor

```
1 SELECT address
2 FROM couchmusic2
3 WHERE email = "delores.riley@hotmail.com";|
```

← history (14/14) →

Execute Explain success | elapsed: 4.16s | execution: 4.16s | count: 1 | size: 235 Preferences

Query Results

JSON Table Tree Plan Plan Text

```
1 [ 
2 { 
3   "address": { 
4     "city": "warren",
5     "countryCode": "US",
6     "postalCode": 63450,
7     "state": "oregon",
8     "street": "6174 elgin st"
9   }
10 }
11 ]
```

Implement a secondary index for a specific document attribute (CREATE INDEX)

2. Create a secondary index for *email* attributes in the *couchmusic2* bucket.

```
CREATE INDEX idx_email ON  
couchmusic2(email);
```

Notice this statement returns with a *success* status, but no values other than the typical query response metrics.

The screenshot shows the Couchbase Query Editor interface. In the 'Query Editor' section, two lines of code are present: 'CREATE INDEX idx\_email ON couchmusic2(email);'. The first line is highlighted with a red box. Below the editor, the status bar shows 'success | elapsed: 3.25s | execution: 3.25s | count: 0 | size: 0'. In the 'Query Results' section, the JSON output is shown: '1: { 2: "results": [] 3: }'. The 'JSON' tab is selected in the results header.

3. Re-run the prior query, selecting *address* attributes by *email* address. You can press the *back-arrow* button in the query history, and press *Execute*.

```
SELECT address FROM  
couchmusic2
```

```
WHERE email = "delores.riley@hotmail.com";
```

Notice the execution time drops to milliseconds.

The screenshot shows the Couchbase Query Editor interface. At the top, there's a toolbar with 'Query Editor' and a history section labeled 'history (14/15)'. Below the toolbar is a code editor containing the following SQL-like query:

```
1 SELECT address
2 FROM couchmusic2
3 WHERE email = "delores.riley@hotmail.com";
```

Below the code editor, there are several buttons: 'Execute' (highlighted with a yellow star), 'Explain', 'success', 'elapsed: 2.09ms | execution: 2.07ms | count: 1 | size: 235', and 'Preferences'. The 'success' button is highlighted with a red box. The 'Query Results' section shows the following JSON output:

```
1 [ 
2 { 
3   "address": { 
4     "city": "warren",
```

Create and use an index on a multiply filtered query (AND)

4. Index the *address.postalCode* attributes of *couchmusic2*.

```
CREATE INDEX idx_postalCode
ON couchmusic2(address.postalCode);
```

5. Select *firstName*, *lastName*, and *address* attributes where the *postalCode* is *63450* and

*firstName* is "Roger".

```
SELECT firstName, lastName, address
FROM couchmusic2
WHERE address.postalCode = 63450
AND firstName = "Roger";
```

Notice the query utilizes the available *address.postalCode* index to provide millisecond response, even though the *firstName* attribute is not indexed.

Query Editor

```
1 SELECT firstName, lastName, address
2 FROM couchmusic2
3 WHERE address.postalCode = 63450
4 AND firstName = "Roger";
```

← history (17/17) →

Execute Explain success | elapsed: 4.31ms | execution: 4.30ms | count: 1 | size: 312 Preferences

Query Results

JSON Table Tree Plan Plan Text

```
1 [ ]
2 [
3   {
4     "address": {
5       "city": "Clane",
6       "countryCode": "IE",
7       "postalCode": 63450,
8       "state": "california",
9       "street": "3587 patrick street"
10    },
11    "firstName": "Roger",
12    "lastName": "Smythe"
13 }
```

Implement an index for an attribute and filter (CREATE INDEX ... WHERE)

6. Index the *address.state* attributes of *couchmusic2* where the *status* attribute of a document is "*active*".

```
CREATE INDEX idx_state_active ON
couchmusic2 (address.state) WHERE
status = "active";
```

7. Select *email* attributes where *address.state* is "*oregon*" (note, the *address.state* values are all lowercase. Case insensitive searches are taught in a later Lab.)

```
SELECT email FROM
couchmusic2
WHERE address.state = "oregon";
```

Notice the index is not used, and it takes several seconds to return 225 documents.

Query Editor

```
1 SELECT email
2 FROM couchmusic2
3 WHERE address.state = "oregon";
```

← history (19/19) →

Execute Explain success | elapsed: 4.16s | execution: 4.16s | count: 225 | size: 13079 Preferences

Query Results

JSON Table Tree Plan Plan Text

```
1- [
2- {
3-   "email": "delores.riley@hotmail.com"
4- },
5- {
6-   "email": "lily.nelson@juno.com"
7- },
8- {
9-   "email": "lesa.west@ymail.com"
```

8. Modify the prior query to filter both by *address.state* and where *status* is "active".

```
SELECT email FROM
couchmusic2
WHERE address.state = "oregon"
AND status = "active";
```

Notice the index is now used to return 164 results in milliseconds.

Query Editor

```
1 SELECT email
2 FROM couchmusic2
3 WHERE address.state = "oregon"
4 AND status = "active";|
```

← history (20/20) →

Execute Explain success | elapsed: 9.87ms | execution: 9.85ms | count: 164 | size: 9509 Preferences

Query Results

JSON Table Tree Plan Plan Text

```
1 [ 
2 {
3   "email": "delores.riley@hotmail.com"
4 },
5 {
```

-----End of Lab-----

## 15. Manipulating data using N1QL DML – 20 Minutes

### Objectives

A	INSERT a new document
B	UPSERT and UPDATE new and existing documents
C	DELETE documents by value or key

Execute the following using Query Workbench : Web console → Query → Query Editor  
INSERT a new userprofile document

1. Insert a new *userprofile* document to *couchmusic2*, using *userprofile::aaa-oscar-orange* as its key, and the values shown below. Note, as expectable, INSERT returns metadata, including execution time, but no results.

```
INSERT INTO couchmusic2 (KEY, VALUE)
VALUES ("userprofile::aaa-oscar-
orange",
{
  "address": {
    "city": "Portland",
```

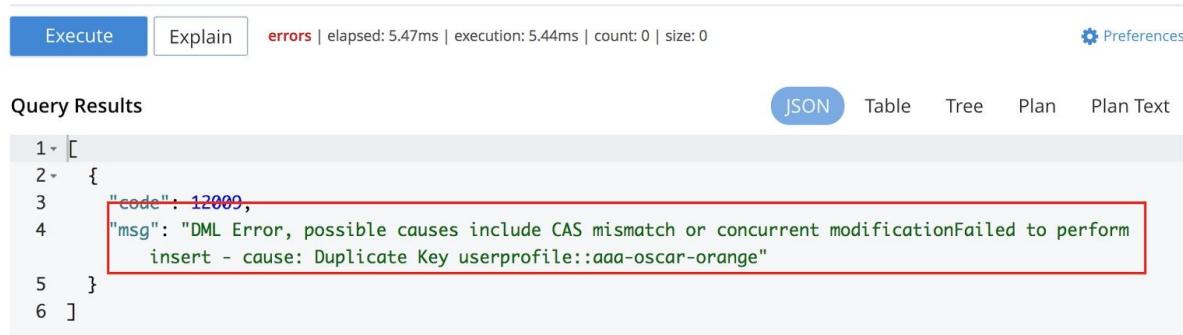
```
"countryCode": "US",
"postalCode": 97203, "state":
"Oregon"
},
"dateOfBirth": "1969-01-23",
"email": "oscar.orange@gmx.com",
"favoriteGenres": [
"Post Punk", "Contemporary
Blues", "Ambient"
],
"firstName": "Oscar",
"gender": "male", "lastName":
"Orange", "phones": [
{
"number": "(503)-555-1222",
"type": "home",
"verified": "2016-05-07T11:11:23"
}
],
"status": "active",
"title": "Mr", "type":
"userprofile",
"updated": "2016-05-23T07:23:32",
```

```

    "username": "aaa-oscar-orange"
}
);

```

2. Execute this query a second time, and notice the duplicate key error.



The screenshot shows a Couchbase query results interface. At the top, there are buttons for 'Execute' (highlighted in blue), 'Explain', and 'errors' (highlighted in red). Below that, status information: elapsed: 5.47ms | execution: 5.44ms | count: 0 | size: 0. On the right, there's a 'Preferences' button. The main area is titled 'Query Results' and has tabs for 'JSON' (highlighted in blue), 'Table', 'Tree', 'Plan', and 'Plan Text'. The JSON output is as follows:

```

1- [
2- {
3-   "code": 12009,
4-   "msg": "DML Error, possible causes include CAS mismatch or concurrent modificationFailed to perform
      insert - cause: Duplicate Key userprofile::aaa-oscar-orange"
5- }
6- ]

```

A red rectangular box highlights the 'msg' field of the JSON response, which contains the error message: "DML Error, possible causes include CAS mismatch or concurrent modificationFailed to perform insert - cause: Duplicate Key userprofile::aaa-oscar-orange".

## UPsert and UPDATE new and existing documents

3. Modify the INSERT statement to an UPSERT statement.
4. Execute the query a third time, and notice it now succeeds.

```

UPSERT INTO couchmusic2 (KEY, VALUE) VALUES ("userprofile::aaa-oscar-
orange",
{
"address": {
"city": "Portland",
"countryCode": "US", "postalCode": 97203, "state": "Oregon"
},

```

```
"dateOfBirth": "1969-01-23", "email": "oscar.orange@gmx.com",
"favoriteGenres": [
    "Post Punk", "Contemporary Blues", "Ambient"
],
"firstName": "Oscar", "gender": "male", "lastName": "Orange", "phones": [
{
    "number": "(503)-555-1222",
    "type": "home",
    "verified": "2016-05-07T11:11:23"
}
],
"status": "active",
"title": "Mr", "type": "userprofile",
"updated": "2016-05-23T07:23:32",
"username": "aaa-oscar-orange"
};
);
```

The screenshot shows the Couchbase Query Editor interface. In the Query Editor pane, a query is displayed:

```
1 UPSERT INTO couchmusic2 (KEY, VALUE)
2 VALUES ("userprofile::aaa-oscar-orange",
3
4     "genre": "Post Punk",
5
6     "name": "Oscar Orange",
7     "age": 25,
8     "city": "Portland",
9     "state": "Oregon",
10    "countryCode": "US",
11    "postalCode": 97203,
12    "address": {
13        "street": "123 Main St",
14        "suite": "Apt A"
15    }
16 )
```

The first two lines of the query are highlighted with a red box. Below the editor, the status bar shows "success | elapsed: 3.93ms | execution: 3.91ms | mutations: 1 | size: 0".

In the Query Results pane, the response is shown as JSON:

```
1 { "results": [] }
```

5. Modify the document attributes as shown in bold below.

```
UPSERT INTO couchmusic2 (KEY, VALUE)
VALUES ("userprofile::aaa-betty-
blue",
{
    "address": {
        "city": "Portland",
        "countryCode": "US",
        "postalCode": 97203, "state":
        "Oregon"
    },
}
```

```
"dateOfBirth": "1969-01-23",
"email": "betty.blue@gmx.com",
"favoriteGenres": [
    "Post Punk", "Contemporary
    Blues", "Ambient"
],
"firstName": "Betty", "gender":
"female",
"lastName": "Blue",
"phones": [
    {
        "number": "(503)-555-1222",
        "type": "home",
        "verified": "2016-05-07T11:11:23"
    }
],
"status": "active",
"title": "Ms",
"type": "userprofile",
"updated": "2016-05-23T07:23:32",
"username": "aaa-betty-blue"
}
);
```

6. Execute the query.
7. Index the *username* attribute of *userprofile* documents in *couchmusic2*.

```
CREATE INDEX idx_userprofile_username  
ON couchmusic2(username)  
WHERE type = "userprofile";
```

8. Select the *firstName*, *lastName*, and *email* attributes of *userprofile* documents where the *username* is *aaa-oscar-orange*.

```
SELECT firstName, lastName, email  
FROM couchmusic2  
WHERE type = "userprofile"  
AND username = "aaa-oscar-orange";
```

9. Update the *email* for this *username*, and update the *updated* timestamp, as shown:

```
UPDATE couchmusic2  
SET email = "oscar.orange2@gmx.com",  
    updated = NOW_STR()  
WHERE type = "userprofile"  
AND username = "aaa-oscar-orange";
```

10. Select all attributes for the *userprofile* with this *username*, to *confirm* the update.

```
SELECT *
FROM couchmusic2
WHERE type = "userprofile"
AND username = "aaa-oscar-orange";
```

DELETE documents by value or key

11. Delete *userprofile* documents with this *username*.

```
DELETE
FROM couchmusic2
WHERE type = "userprofile"
AND username = "aaa-oscar-orange";
```

12. Delete *userprofile* documents which have the key *userprofile::aaa-betty-blue*.

```
DELETE
FROM couchmusic2
USE KEYS ("userprofile::aaa-betty-blue");
```

13. In the Data Buckets UI, verify the Userprofiles added in this lab have been deleted.

The screenshot shows the 'Data Buckets' interface for a bucket named 'couchmusic2'. A search bar at the top contains the filter query: `?startkey=%22userprofile%22&skip=0&include_docs=true&limit=11`. Below the filter, there are fields for 'startkey' (set to 'userprofile') and 'endkey' (set to an empty string). A checkbox for 'inclusive\_end' is unchecked. At the bottom right are 'Reset' and 'Save' buttons, with 'Save' being highlighted in blue.

-----End of Lab -----

**16. N1QL Explain-Querying ranges, ordering results, and explaining query details -30 Minutes****Objectives**

A	Query a value range and order the results (AND, ORDER BY)
B	Introspect (EXPLAIN) a query to determine its index use
C	Verify available indexes in the Couchbase console

Query a value range and order the results (AND, ORDER BY)

1. From the *CB110-Data/samples* folder, open the *country.json* document. Review the *Country* document attributes and their structure.

```
country.json *  
1 {  
2   "gdp": 40214,  
3   "updated": "2015-10-01T07:35:13",  
4   "region-number": 39,  
5   "name": "Andorra",  
6   "countryCode": "AD",  
7   "type": "country",  
8   "population": 80792  
9 }
```

2. Select the *name* and *population* attributes of *country* documents, only, where the *population* is between one and five million, inclusive.

```
SELECT name, population FROM
couchmusic2
WHERE population >= 1000000 AND population <=
5000000 AND type = "country";
```

Notice the documents appear ordered by name. This is not guaranteed behavior.

Query Results

JSON Table Tree Plan Plan Text

```
1> [
2>   {
3>     "name": "Albania",
4>     "population": 3195196
5>   },
6>   {
7>     "name": "Armenia",
8>     "population": 2989427
9>   },
10>  {
11>    "name": "Bosnia and Herzegovina",
12>    "population": 3816291
13>  }
```



3. Modify the query to sequence the results in descending order, by population.

```
SELECT name, population FROM
couchmusic2
WHERE population >= 1000000 AND population <=
5000000 AND type = "country"
ORDER BY population DESC;
```

Query Results

JSON    Table    Tree    Plan    Plan Text

```
1- [
2-   {
3-     "name": "Central African Republic",
4-     "population": 4806375
5-   },
6-   {
7-     "name": "Ireland",
8-     "population": 4732269
9-   },
10-  {
11-    "name": "Congo",
12-    "population": 4679663
}
```



Introspect (EXPLAIN) a query to determine its index use

4. Click the *Explain* button, examine the execution plan for this query.

Note the `#operator` value, which in this case is `PrimaryScan`, indicating that the primary index will be used to perform a full bucket scan to complete this query.

Query Editor ← history (22/22) →

```
1 SELECT name, population
2 FROM couchmusic2
3 WHERE population >= 1000000 AND population <= 5000000
4 AND type = "country"
5 ORDER BY population DESC;
```

**Execute** **Explain** Explain success | elapsed: | execution: | count: | size: Preferences

Query Results JSON Table Tree Plan Plan Text

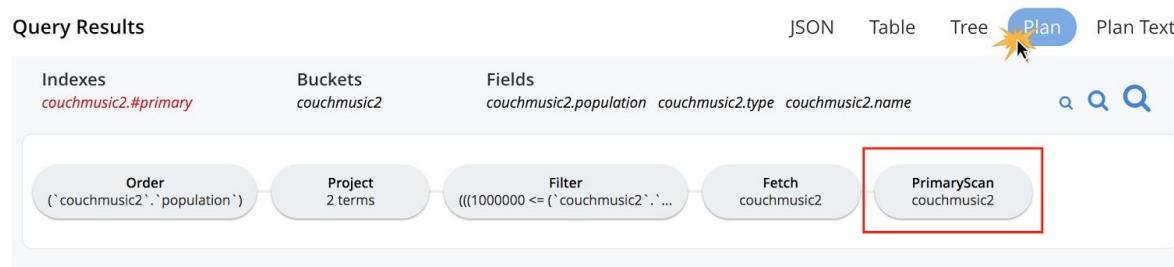
```
1 [
2   {
3     "plan": {
4       "#operator": "Sequence",
5       "~children": [
6         {
7           "#operator": "Sequence",
8           "~children": [
9             {
10               "#operator": "PrimaryScan",
11               "index": "#primary",
12               "keyspace": "couchmusic2",
13               "namespace": "default",
14               "using": "gsi"
15             },
16             {
17               "#operator": "Fetch",
18             }
19           ]
20         }
21       ]
22     }
23   }
24 ]
```

5. (Optional) Alternately, you can modify and re-execute the query to EXPLAIN its execution plan.

### **EXPLAIN**

```
SELECT name, population FROM
couchmusic2
WHERE population >= 1000000 AND population <=
5000000 AND type = "country"
ORDER BY population DESC;
```

6. Examine the Plan Text and Plan for this query. Notice that the visualized Plan identifies that a PrimaryScan will be used for this query.



7. Index the population attribute of country documents, only.

```
CREATE INDEX idx_country_population
ON couchmusic2(population)
WHERE type = "country";
```

8. Re-execute the prior query, and re-examine its Plan. #operator is now an *IndexScan2*.

Query Editor

← history (22/23) →

```
1 SELECT name, population
2 FROM couchmusic2
3 WHERE population >= 1000000 AND population <= 5000000
4 AND type = "country"
5 ORDER BY population DESC;
```

Execute 
Explain 
success | elapsed: 6.03ms | execution: 6.01ms | count: 38 | size: 3017


Query Results

JSON
Table
Tree
Plan 
Plan Text



The diagram illustrates the execution plan for the query. It consists of several stages connected by arrows:

- Order**: `(`couchmusic2`.`population`<`) 00:00:0001 (4.8%) 38 in / 38 out`
- Project**: `2 terms 00:00:0004 (11.1%) 38 in / 38 out`
- Filter**: `((1000000 <=(`couchmusic2`.`... 00:00:0005 (16.7%) 38 in / 38 out`
- Fetch**: `couchmusic2 00:00:0011 (33%) 38 in / 38 out`
- IndexScan2**: `00:00:0005 (16.4%) 38 out` (This stage is highlighted with a red box.)
- Authorize**: `00:00:0006 (16.9%)`

6. Review the full structure of the Plan Text. Notice the *index* in use is identified, along with *spans*, indicating the *high* and *low range* being filtered. Notice the bucket name is referred to as *keyspace*.

Query Results

JSON   Table   Tree   Plan   **Plan Text**

```

42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
  "#operator": "IndexScan2",
  "#stats": {
    "#itemsOut": 38,
    "#phaseSwitches": 155,
    "execTime": "61.125µs",
    "kernTime": "17.673µs",
    "servTime": "528.32µs"
  },
  "index": "idx_country_population",
  "index_id": "a708e18fadb9d90ff",
  "index_projection": {
    "primary_key": true
  },
  "keyspace": "couchmusic2",
  "namespace": "default",
  "spans": [
    {
      "exact": true,
      "range": [
        {
          "high": "5000000",
          "inclusion": 3,
          "low": "1000000"
        }
      ]
    }
  ]
}

```

Obviously, there is much more to the Plan text.

Verify available indexes in the Couchbase console and by query

4. Index the *status* attribute of *userprofile* documents in *couchmusic2*.

```
CREATE INDEX idx_userprofile_status
ON couchmusic2(status)
```

```
WHERE type = "userprofile";
```

5. In the Couchbase console, open the Indexes UI for Global Indexes, and verify the

The screenshot shows the Couchbase Admin UI with the 'Indexes' tab selected (indicated by a red circle). The main table lists global indexes for the 'couchmusic2' bucket. One index, 'idx\_userprofile\_status', is highlighted with a red box. A tooltip below it shows its definition:

```
Definition  
CREATE INDEX `idx_userprofile_status` ON `couchmusic2`(`status`) WHERE (`type` = "userprofile")
```

*idx\_userprofile\_status* index has been created. Click to examine its text.

6. In the Query Workbench, query all *name*, *index\_key*, and *condition* attributes in the *system:indexes* keyspace to view these attributes of all 6 indexes you have created for *couchmusic2*. Notice which have *condition* attributes, and consider why this is so.

```
SELECT name, index_key, condition
FROM system:indexes
WHERE keyspace_id = "couchmusic2";
```

Query Editor

```
1 SELECT name, index_key, condition
2 FROM system:indexes
3 WHERE keyspace_id = "couchmusic2";|
```

← history (25/25) →

Execute Explain success | elapsed: 42.53ms | execution: 42.51ms | count: 6 | size: 857 Preferences

Query Results

JSON Table Tree Plan Plan Text

```
1 [
2 {
3   "condition": "(`type` = \"userprofile\")",
4   "index_key": [
5     "`status`"
6   ],
7   "name": "idx_userprofile_status"
8 },
9 {
10   "condition": "(`type` = \"country\")",
11   "index_key": [
12     "`population`"
13   ],
14   "name": "idx_country_population"
```

-----End of Lab-----

## 17. Tuning Query – (Explain to determine query plan) – 30 Minutes

Execution of a N1QL query by the engine involves multiple steps. Understanding these will help you to write queries, design for performance, tune query engine efficiently. The N1QL query engine includes parser, semantic analyzer, optimizer and executor.

Every query has a query plan. The performance and efficiency of a query depend on its plan. The N1QL query engine prepares the query plan for each query. N1QL uses a rule-based optimizer. The creation and selection of the right index have a major influence on both performance and efficiency. This article will walk through the step-by-step process of how to create an index and modify the query for optimal performance.

We'll use travel-sample dataset shipped with Couchbase. [Install travel-sample](#) shipped with Couchbase, then drop all of the indexes to start with a clean slate.

[Note Adding Sample Bucket : Click on Setting → Sample Bucket ( Right Top corner) → Select travel-sample → Load Sample Data]

The screenshot shows the Couchbase Web Console with the URL [Tos.Couchbase > Settings](#). The top navigation bar includes links for Cluster, Software Updates, Auto-Failover, Email Alerts, Auto-Compaction, and Sample Buckets (with a dropdown arrow). The left sidebar lists various management options: Dashboard, Servers, Buckets, Indexes, Search, Query, XDCR, Security, and Settings. The main content area is titled "Sample Buckets" and contains the following information:

- Dashboard:** Sample buckets contain example data and Couchbase views.
- Servers:** You can provision one or more sample buckets to help you discover the power of Couchbase Server.
- Buckets:** Sample buckets (like all buckets in Couchbase Server 5.0+) can only be accessed by a user with privileges for that bucket.
- Indexes:** Available Samples: beer-sample, gamesim-sample, travel-sample. Installed Samples: none.
- Search:**
- Query:**
- XDCR:**
- Security:**
- Settings:** A blue "Load Sample Data" button.

You can verify the bucket from the Web console using Buckets option.

The screenshot shows the Couchbase Web Console with the URL [Tos.Couchbase > Buckets](#). The top navigation bar includes links for Activity, Documentation, Support, and Administrator. The left sidebar lists management options: Dashboard, Servers, Buckets, and Indexes. The main content area displays a table of buckets:

name ▾	items	resident	ops/sec	RAM used/quota	disk used	Documents	Statistics
default	100,000	100%	0	20.6MB / 100MB	18.9MB	<a href="#">Documents</a>	<a href="#">Statistics</a>
travel-sample	31,591	100%	0	62.6MB / 100MB	50.1MB	<a href="#">Documents</a>	<a href="#">Statistics</a>

Verify all the indexes created for the travel-sample bucket. You can perform this activity using the following option:

Click on Indexes option in Web console.

	Global Indexes ▾		Views			
	bucket ▾	node	index name	storage type	status	build progress
Indexes	travel-sample	10.10.20.28:8091	def_airportname	Standard GSI	ready	100%
	travel-sample	10.10.20.28:8091	def_city	Standard GSI	ready	100%
	travel-sample	10.10.20.28:8091	def_faa	Standard GSI	ready	100%
	travel-sample	10.10.20.28:8091	def_icao	Standard GSI	ready	100%
	travel-sample	10.10.20.28:8091	def_name_type	Standard GSI	ready	100%
	travel-sample	10.10.20.28:8091	def_primary	Standard GSI	ready	100%
	travel-sample	10.10.20.28:8091	def_route_src_dst_day	Standard GSI	ready	100%
	travel-sample	10.10.20.28:8091	def_schedule_utc	Standard GSI	ready	100%
	travel-sample	10.10.20.28:8091	def_sourceairport	Standard GSI	ready	100%
	travel-sample	10.10.20.28:8091	def_type	Standard GSI	ready	100%

Let us drop all the above indexes. Enter the following query in the Query Workbench one at a time.  
**Query → Query Editor → Execute**

```
DROP INDEX `travel-sample`.def_city;
DROP INDEX `travel-sample`.def_faa;
DROP INDEX `travel-sample`.def_icao;
DROP INDEX `travel-sample`.def_name_type;
DROP INDEX `travel-sample`.def_airportname;
DROP INDEX `travel-sample`.def_schedule_utc;
DROP INDEX `travel-sample`.def_type;
DROP INDEX `travel-sample`.def_sourceairport;
DROP INDEX `travel-sample`.def_route_src_dst_day;
```

```
DROP INDEX `travel-sample`.def_primary;
```

E.x

The screenshot shows the Couchbase Query Editor interface. At the top, there is a history bar with a left arrow, the text "history (11/11)", and a right arrow. Below the history bar is a code editor containing the command: `1 DROP INDEX `travel-sample`.def_primary;`. Below the code editor are two buttons: "Execute" (highlighted in blue) and "Explain". To the right of these buttons, the status is shown as "success | elapsed: 77.69ms | execution: 77.65ms | count: 0 | size: 0". Further to the right is a "Preferences" button. Below the code editor, the title "Query Results" is followed by a "JSON" button (which is highlighted in blue), and other options: "Table", "Tree", "Plan", and "Plan Text". The results pane displays the JSON response: `1 {  
2 "results": []  
3 }`.

We will walk through the step-by-step process and create the index for the following query. In this query, we're looking to get airlines in the United States with an id between zero and 1000, ordered by id. We're interested in the 10 airlines, ordered by id and skipping the first five.

```
SELECT country, id, name
FROM `travel-sample`
WHERE type = "airline"
AND country = "United States"
AND id BETWEEN 0 AND 1000
ORDER BY id
LIMIT 10
OFFSET 5;
```

Try executing the above query.

The screenshot shows the 'Query Editor' tab in the Couchbase Query Workbench. The query is:

```

1 SELECT country, id, name
2 FROM `travel-sample`
3 WHERE type = "airline"
4 AND country = "United States"
5 AND id BETWEEN 0 AND 1000
6 ORDER BY id
7 LIMIT 10
8 OFFSET 5;

```

The status bar indicates a 404 error with an elapsed time of 5.29ms and an execution time of 5.26ms. The 'Execute' button is highlighted.

The 'Query Results' section shows a JSON response:

```

1 [
2 {
3   "code": 4000,
4   "msg": "No index available on keyspace travel-sample that matches your query. Use CREATE INDEX or
          CREATE PRIMARY INDEX to create an index, or check that your expected index is online.",
5   "query_from_user": "SELECT country, id, name\r\nFROM `travel-sample`\r\nWHERE type = \"airline\"\r\nAND
          country = \"United States\"\r\nAND id BETWEEN 0 AND 1000\r\nORDER BY id\r\nLIMIT 10\r\nOFFSET 5;"
6 }
]

```

As shown above you can't execute any query until there is primary index created on that particular bucket.

```
CREATE PRIMARY INDEX ON `travel-sample`;
```

After the creation of primary Index, you can execute the earlier query.

Query Editor

```

1 SELECT country, id, name
2 FROM `travel-sample`
3 WHERE type = "airline"
4 AND country = "United States"
5 AND id BETWEEN 0 AND 1000
6 ORDER BY id
7 LIMIT 10
8 OFFSET 5;

```

**Execute** **Explain** success | elapsed: 3.08s | execution: 3.08s | count: 10 | size: 1153 **Preferences**

Query Results

JSON **Table** Tree Plan Plan Text

country	id	name
United States	149	Air Cargo Carriers
United States	210	Airlift International
United States	281	America West Airlines
United States	282	Air Wisconsin
United States	287	Allegheny Commuter Airlines

The primary index is the simplest index, indexing all of the documents in `travel-sample`. Any query can exploit the primary index to execute the query.

Click on the label, **Table** to display the result in the table format.

Now click on Explain, then Plan,



It shows that, this query used the primary index to fetch the data.

Some of the Main Operations you can observe from the query plan are as shown below:

**PrimaryScan:** Scan of the Primary Index based on document keys

```
{
  "#operator": "Sequence",
  "~children": [
    {
      "#operator": "PrimaryScan",
      "index": "#primary",
      "keyspace": "travel-sample",
      "namespace": "default",
      "using": "gsi"
    }
  ]
}
```

**Fetch:** Reach into the Data service with a document key

```
{
  "#operator": "Fetch",
  "keyspace": "travel-sample",
  "namespace": "default"
},
```

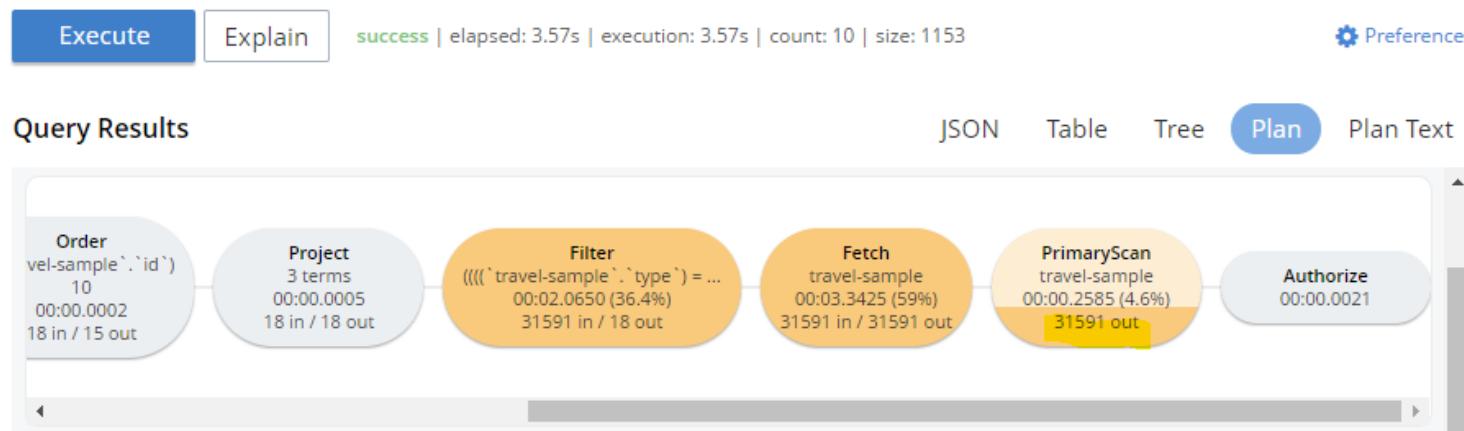
**InitialProject:** reducing the stream size to the fields involved in query.

```
{
  "#operator": "InitialProject",
  "result_terms": [
    {
      "expr": "(`travel-sample`.`country`)"
    },
    {
      "expr": "(`travel-sample`.`id`)"
    },
    {
      "expr": "(`travel-sample`.`name`)"
    }
  ]
}
```

**Parallel:** execute all child operations in parallel

```
{
  "#operator": "Parallel",
  "~child": {
    "#operator": "Sequence",
    "~children": [
      {
        "#operator": "Filter",
        "condition": "(((`travel-sample`.`type` = \"airline\") and ((`travel-sample`.`country` = \"USA\"))) or ((`travel-sample`.`type` = \"train\") and ((`travel-sample`.`country` = \"China\"))))"
      }
    ]
  }
}
```

Now click on the Plan after executing the query.



You can clearly see from above that it scan all the documents in the bucket i.e 31591 for this particular query.

Primary Index can only filter on document keys thus typically means “full-scan” of the bucket. Secondary Index is typically done with predicates and are smaller in size thus better to scan.  
Let us create a secondary index and determine the Query Plan.

CREATE INDEX i\_country on `travel-sample`(country) USING GSI;

Query Editor

```
1 CREATE INDEX i_country ON `travel-sample`(country) USING GSI;
```

← history (17/17) →

Execute Explain success | elapsed: 5.73s | execution: 5.73s | count: 0 | size: 0 Preferences

Query Results

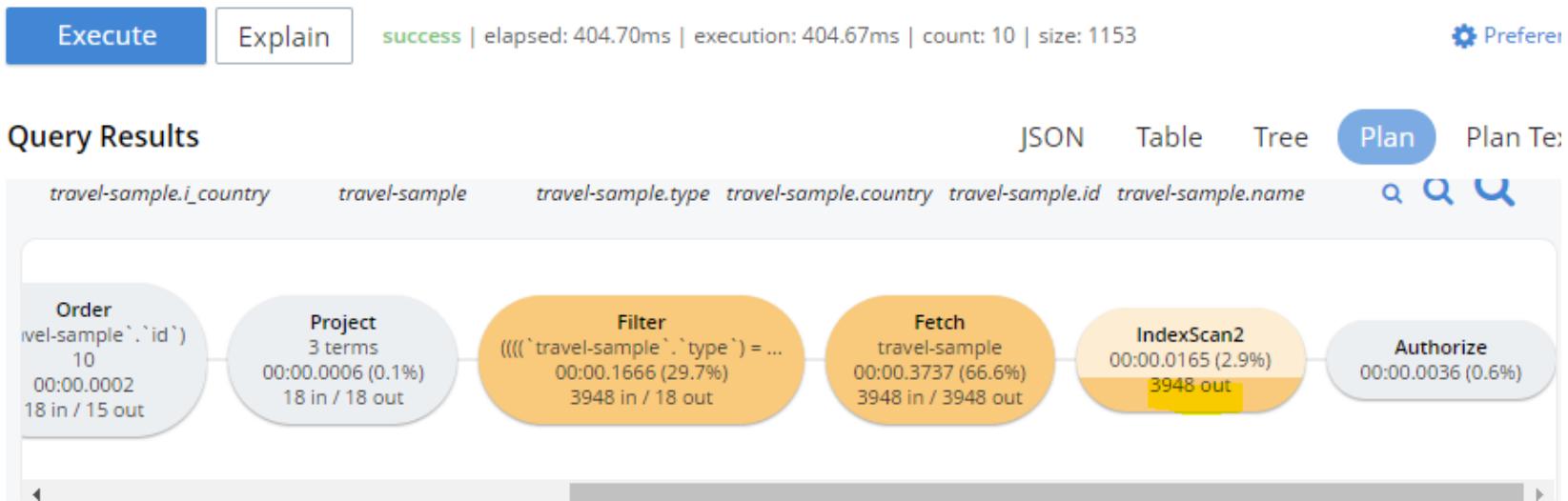
Indexes Buckets Fields

travel-sample.i\_country travel-sample

Stream 00:00.0000 CreateIndex 00:05.7220 (99.9%) Authorize 00:00.0030 (0.1%)

The screenshot shows the Couchbase Query Editor interface. In the Query Editor section, a single line of code is entered: 'CREATE INDEX i\_country ON `travel-sample`(country) USING GSI;'. Below the editor, a status bar indicates 'success' with metrics: elapsed: 5.73s, execution: 5.73s, count: 0, and size: 0. There are 'Execute' and 'Explain' buttons, along with a 'Preferences' link. The 'Explain' button is currently selected. In the 'Query Results' section, tabs for 'JSON', 'Table', 'Tree', 'Plan', and 'Plan Text' are available, with 'Plan' being the active tab. The 'Plan' view displays the execution plan for the query, showing three stages: 'Stream' (00:00.0000), 'CreateIndex' (00:05.7220, 99.9%), and 'Authorize' (00:00.0030, 0.1%). The 'CreateIndex' stage is highlighted with an orange background.

Execute the same query again.



Now its secondary scan only with 3948

----- Lab Ends Here -----

**18. XDCR – 90 Minutes (D)**

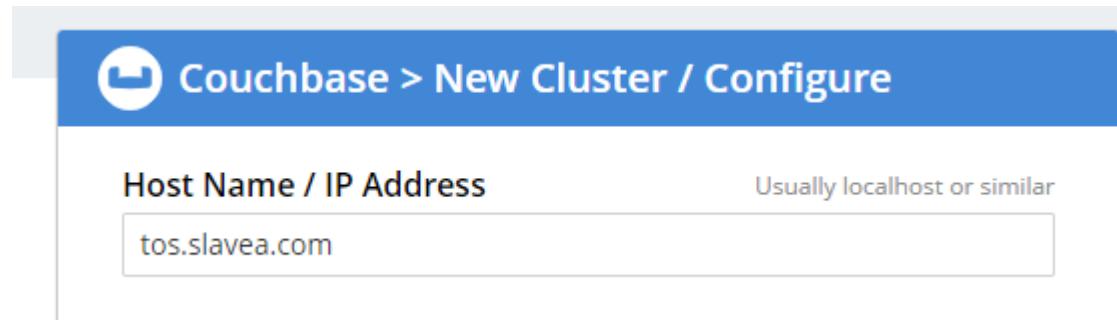
You need two cluster for this lab:

Use CLI Slave A as the Cluster II and CLI Master as Cluster I.

Hostname : tos.slavea.com

Install Couchbase on Slave A VM. You can refer the Installation Lab for that.

Ensure that Cluster Name as : Tos.CouchbaseII in this case. Rest of the setting will remain the same.

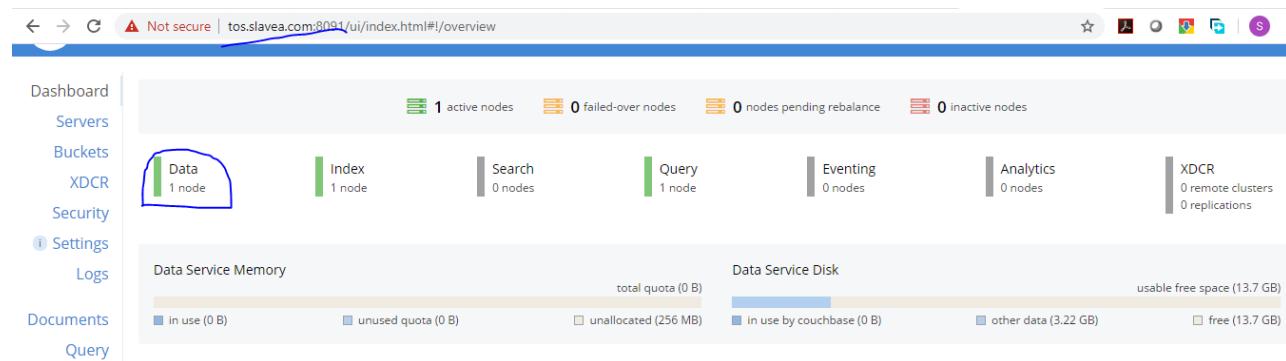


### Service Memory Quotas

Per service / per node

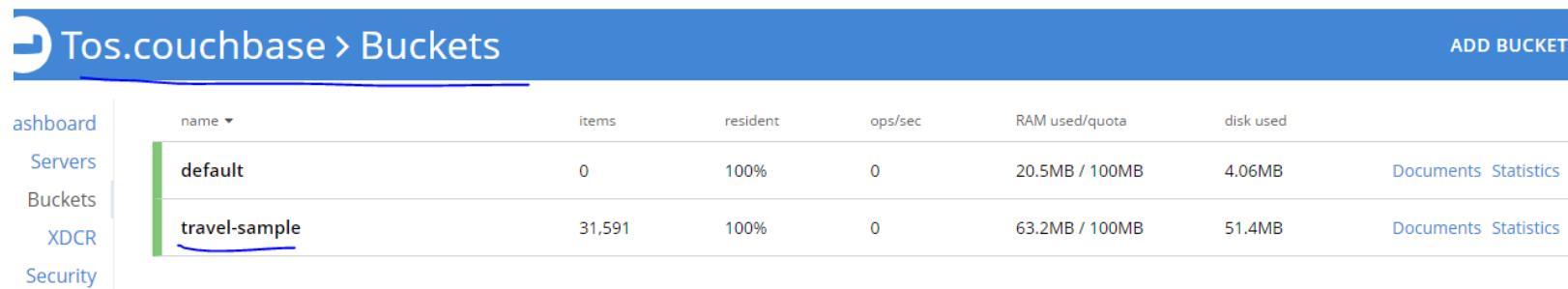
<input checked="" type="checkbox"/> Data	256	MB
<input checked="" type="checkbox"/> Index	512	MB
<input type="checkbox"/> Search	256	MB
<input checked="" type="checkbox"/> Query	-----	
<input type="checkbox"/> Eventing	256	MB
<input type="checkbox"/> Analytics	1024	MB

At the end of installation:



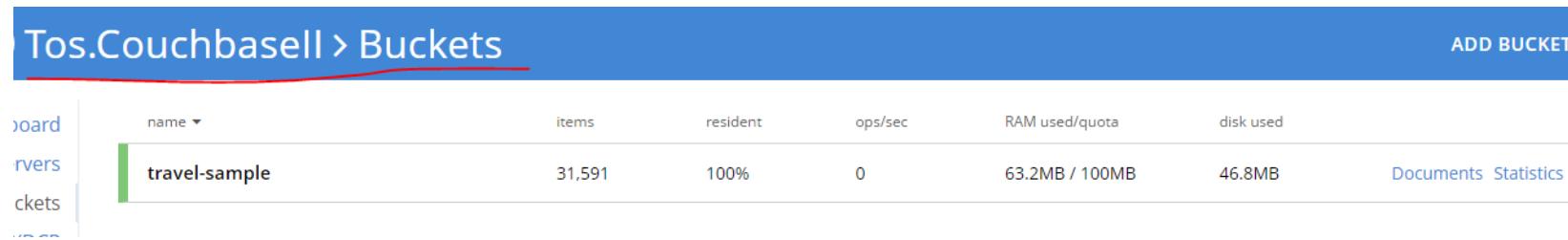
Once your source and target clusters have been prepared, to start XDCR management, ensure that

Each cluster contains a single bucket, which is the travel-sample bucket. To access and install this, see [Sample Buckets](#)



The screenshot shows the 'Buckets' section of the Tos.couchbase interface. On the left, there is a navigation sidebar with links forashboard, Servers, Buckets (which is selected and highlighted in green), XDCR, and Security. The main content area has a header 'Tos.couchbase > Buckets' and an 'ADD BUCKET' button. A table lists two buckets:

name ▾	items	resident	ops/sec	RAM used/quota	disk used	Documents	Statistics
default	0	100%	0	20.5MB / 100MB	4.06MB	<a href="#">Documents</a>	<a href="#">Statistics</a>
travel-sample	31,591	100%	0	63.2MB / 100MB	51.4MB	<a href="#">Documents</a>	<a href="#">Statistics</a>



The screenshot shows the 'Buckets' section of the Tos.Couchbasell interface. On the left, there is a navigation sidebar with links forashboard, Servers, Buckets (selected and highlighted in green), XDCR, and Security. The main content area has a header 'Tos.Couchbasell > Buckets' and an 'ADD BUCKET' button. A table lists one bucket:

name ▾	items	resident	ops/sec	RAM used/quota	disk used	Documents	Statistics
travel-sample	31,591	100%	0	63.2MB / 100MB	46.8MB	<a href="#">Documents</a>	<a href="#">Statistics</a>

## Create an XDCR Reference with the UI

Proceed as follows: Using  
Tos.couchbase Cluster Console.

1. Access Couchbase Web Console. Left-click on the XDCR tab, in the left-hand navigation menu.



This displays the XDCR Replications screen:

The screenshot shows the Couchbase Admin UI with the following details:

- Header:** Activity Documentation Support Administrator ▾
- Page Title:** Tos.couchbase > XDCR Replications
- Left Sidebar:** Dashboard, Servers, Buckets, XDCR, Security, **Settings**, Logs, Documents, Query, Search.
- Main Content:**
  - Remote Clusters:** A table with columns name and IP/hostname. A message says "No cluster references defined."
  - Ongoing Replications:** A table with columns bucket, protocol, from, to, filtered, status, and when. A message says "There are no replications currently in progress."
- Right Top:** Add Remote Cluster button.

The lower part of the main panel is entitled Remote Clusters. The list, which is designed to show the name and IP address or hostname of each registered remote cluster, is currently empty, and so bears the notification No cluster references defined.

2. Define a reference, by left-clicking on the Add Remote Cluster button, at the upper right.



The Add Remote Cluster dialog is now displayed:

Add Remote Cluster X

Cluster Name

IP/Hostname ⓘ

Username for Remote Cluster

Password

Enable Secure Connection ⓘ

[Cancel](#) [Save](#)

3. For Cluster Name (Second Cluster name) and IP/Hostname, specify the IP address of the second cluster, which is `tos.slavea.com`. For Username and Password, specify those stated above. Do not, for the current example, check the Enable Secure Connection checkbox. The complete dialog appears as follows:

Add Remote Cluster X

Cluster Name  
TOS.Couchbasell

IP/Hostname ⓘ  
tos.slavea.com

Username for Remote Cluster  
Administrator

Password  
.....|

Enable Secure Connection ⓘ

Cancel Save

When you have entered the data, left-click on the Save button.  
The XDCR Replications screen is again displayed. The Remote Clusters panel now contains the reference you have defined.

## .couchbase > XDCR Replications

Remote Clusters		Add Remote Cluster
name	IP/hostname	
TOS.Couchbasell	tos.slavea.com:8091	<a href="#">Delete</a> <a href="#">Edit</a>

This concludes reference-definition.

Then after reference Create an XDCR Replication with the UI

Proceed as follows: Cluster Tos.Couchbase

1. Access Couchbase Web Console. Left-click on the XDCR tab, in the left-hand navigation menu.



This displays the XDCR Replications screen, the lower part of the main panel of which is entitled Ongoing Replications:

couchbase > XDCR Replications

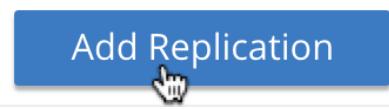
Remote Clusters		Add Remote Cluster
name	IP/hostname	
TOS.Couchbase11	tos.slavea.com:8091	Delete Edit

Ongoing Replications

bucket	protocol	from	to	filtered	status	when
There are no replications currently in progress.						

The list, which is designed to show the name and IP address or hostname of each existing replication, is currently empty, and so bears the notification There are no replications currently in progress.

2. To start creating a replication, left-click on the Add Replication button:



Add Replication

This brings up the Add Replication dialog:

**Add Replication**

**Replicate From Bucket**  
select a bucket

**Remote Cluster**  
Pick remote cluster

**Remote Bucket**  
[empty input field]

**XDCR Protocol**  
Version 2

Enable advanced filtering

► Show Advanced Settings

Cancel Save

Or

**Dashboard**      **Replicate From Bucket**      **Remote Bucket**      **Remote Cluster**      **Mapping Rules**

Servers      Learning      learning      tos.slave.com

Buckets

Backup

XDCR

Security

Settings

Logs

Documents

The bucket **Learning** will be replicated to the bucket **learning** on the remote cluster **tos.slave.com**.  
Missing targets on the remote will be ignored and that data will not be replicated.

Filter Replication

Specify Scopes, Collections, and Mappings

Use Migration Mode ⓘ

► Advanced Settings

3. Enter appropriate information into the fields of the Add Replication dialog.

Specify Secondary cluster as the target cluster, and travel-sample as both source and target bucket. Leave the XDCR Protocol as Version 2. At this stage, do not check the Enable advanced filtering checkbox, and do not elect to Show Advanced Settings.

The completed dialog now appears as follows.

## Add Replication

X

Replicate From Bucket

travel-sample ▾

Remote Cluster

TOS.Couchbasell ▾

Remote Bucket

travel-sample|

XDCR Protocol

Version 2 ▾

Enable advanced filtering

▶ Show Advanced Settings

Cancel Save

Left-click on the Save button. The XDCR Replications screen is now redisplayed, with the appearance of the Ongoing Replications panel as follows:

Ongoing Replications							Add Replication
bucket	protocol	from	to	filtered	status	when	
travel-sample	Version 2	this cluster	bucket "travel-sample" on cluster "TOS.Couchbasell"	No	Replicating 		<a href="#">Delete</a> <a href="#">Edit</a>

This indicates that a replication is now in progress: from travel-sample on this cluster to bucket "travel-sample" on cluster "TOS.Couchbasell".

This concludes creation of the replication.

#### Advanced Settings with the UI Click on Edit

Advanced Settings can be established by left-clicking on the Show Advanced Settings control, on the Add Replication dialog. The UI expands vertically, to reveal the following:

▼ Show Advanced Settings

**XDCR Compression Type****XDCR Source Nozzles Per Node****XDCR Target Nozzles Per Node****XDCR Checkpoint Interval****XDCR Batch Count****XDCR Batch Size (kB)****XDCR Failure Retry Interval****XDCR Optimistic Replication Threshold****XDCR Statistics Collection Interval (ms)****XDCR Network Usage Limit (MB/sec)****XDCR Logging Level**[Cancel](#)[Save](#)

The values displayed in the fields are defaults, which can be modified interactively, and saved: this may help in achieving optimal replication-performance.

Changes the Compression type as shown below:

Advanced Settings X

XDCR Compression Type

Snappy ▾

And Save it

Update a record on the Source Cluster. i.e TOS.Couchbase and let us verify that the changes are reflected on the target cluster i.e TOS.CouchbaseII

Click On Buckets → Document button of travel-sample and edit option as shown below.

Bucket travel-sample Limit 10 Offset 0 Document ID Where Retrieve Docs

10 Results for travel-sample, limit: 10, offset: 0 simple spreadsheet < Prev Batch Next Batch >

	id	Document
<input checked="" type="checkbox"/>	airline_10	{"callsign":"MILE-AIR", "country":"United States", "iata":"Q5", "icao":"MLA", "id":10, "name":"40-Mile Air", "type":"airline"}

Changes the country from USA to India.

Edit Document X

airline\_10 metadata

```
1 > {
2     "callsign": "MILE-AIR",
3     "country": "India",
4     "iata": "Q5",
5     "icao": "MLA",
6     "id": 10,
7     "name": "40-Mile Air",
8     "type": "airline"
9 }
```

Save the document.

Go the web console of TOS.Couchbase II and verify the record as shown below:

The screenshot shows the Couchbase UI interface. On the left, there's a sidebar with a 'Bucket' dropdown set to 'travel-sample'. Below it, a message says '10 Results for travel-sample, limit 10'. A list of documents is shown with icons for each row. In the center, a modal window titled 'Edit Document' is open for a document named 'airline\_10'. The document content is displayed as JSON:

```
1 > {  
2   "callsign": "MILE-AIR",  
3   "country": "India",  
4   "iata": "Q5",  
5   "icao": "MLA",  
6   "id": 10,  
7   "name": "40-Mile Air",  
8   "type": "airline"  
9 }  
10 > {  
11   "callsign": "TXW",  
12   "country": "India",  
13   "iata": "A1F",  
14   "icao": "IPB",  
15   "id": 11,  
16   "name": "40-Mile Air",  
17   "type": "airline"  
18 }  
19 > {  
20   "callsign": "IPB",  
21   "country": "India",  
22   "iata": "A1F",  
23   "icao": "IPB",  
24   "id": 12,  
25   "name": "40-Mile Air",  
26   "type": "airline"  
27 }  
28 > {  
29   "callsign": "IPB",  
30   "country": "India",  
31   "iata": "A1F",  
32   "icao": "IPB",  
33   "id": 13,  
34   "name": "40-Mile Air",  
35   "type": "airline"  
36 }  
37 > {  
38   "callsign": "IPB",  
39   "country": "India",  
40   "iata": "A1F",  
41   "icao": "IPB",  
42   "id": 14,  
43   "name": "40-Mile Air",  
44   "type": "airline"  
45 }  
46 > {  
47   "callsign": "IPB",  
48   "country": "India",  
49   "iata": "A1F",  
50   "icao": "IPB",  
51   "id": 15,  
52   "name": "40-Mile Air",  
53   "type": "airline"  
54 }  
55 > {  
56   "callsign": "IPB",  
57   "country": "India",  
58   "iata": "A1F",  
59   "icao": "IPB",  
60   "id": 16,  
61   "name": "40-Mile Air",  
62   "type": "airline"  
63 }  
64 > {  
65   "callsign": "IPB",  
66   "country": "India",  
67   "iata": "A1F",  
68   "icao": "IPB",  
69   "id": 17,  
70   "name": "40-Mile Air",  
71   "type": "airline"  
72 }  
73 > {  
74   "callsign": "IPB",  
75   "country": "India",  
76   "iata": "A1F",  
77   "icao": "IPB",  
78   "id": 18,  
79   "name": "40-Mile Air",  
80   "type": "airline"  
81 }  
82 > {  
83   "callsign": "IPB",  
84   "country": "India",  
85   "iata": "A1F",  
86   "icao": "IPB",  
87   "id": 19,  
88   "name": "40-Mile Air",  
89   "type": "airline"  
90 }  
91 > {  
92   "callsign": "IPB",  
93   "country": "India",  
94   "iata": "A1F",  
95   "icao": "IPB",  
96   "id": 20,  
97   "name": "40-Mile Air",  
98   "type": "airline"  
99 }  
100 > {  
101   "callsign": "IPB",  
102   "country": "India",  
103   "iata": "A1F",  
104   "icao": "IPB",  
105   "id": 21,  
106   "name": "40-Mile Air",  
107   "type": "airline"  
108 }  
109 > {  
110   "callsign": "IPB",  
111   "country": "India",  
112   "iata": "A1F",  
113   "icao": "IPB",  
114   "id": 22,  
115   "name": "40-Mile Air",  
116   "type": "airline"  
117 }  
118 > {  
119   "callsign": "IPB",  
120   "country": "India",  
121   "iata": "A1F",  
122   "icao": "IPB",  
123   "id": 23,  
124   "name": "40-Mile Air",  
125   "type": "airline"  
126 }  
127 > {  
128   "callsign": "IPB",  
129   "country": "India",  
130   "iata": "A1F",  
131   "icao": "IPB",  
132   "id": 24,  
133   "name": "40-Mile Air",  
134   "type": "airline"  
135 }  
136 > {  
137   "callsign": "IPB",  
138   "country": "India",  
139   "iata": "A1F",  
140   "icao": "IPB",  
141   "id": 25,  
142   "name": "40-Mile Air",  
143   "type": "airline"  
144 }  
145 > {  
146   "callsign": "IPB",  
147   "country": "India",  
148   "iata": "A1F",  
149   "icao": "IPB",  
150   "id": 26,  
151   "name": "40-Mile Air",  
152   "type": "airline"  
153 }  
154 > {  
155   "callsign": "IPB",  
156   "country": "India",  
157   "iata": "A1F",  
158   "icao": "IPB",  
159   "id": 27,  
160   "name": "40-Mile Air",  
161   "type": "airline"  
162 }  
163 > {  
164   "callsign": "IPB",  
165   "country": "India",  
166   "iata": "A1F",  
167   "icao": "IPB",  
168   "id": 28,  
169   "name": "40-Mile Air",  
170   "type": "airline"  
171 }  
172 > {  
173   "callsign": "IPB",  
174   "country": "India",  
175   "iata": "A1F",  
176   "icao": "IPB",  
177   "id": 29,  
178   "name": "40-Mile Air",  
179   "type": "airline"  
180 }  
181 > {  
182   "callsign": "IPB",  
183   "country": "India",  
184   "iata": "A1F",  
185   "icao": "IPB",  
186   "id": 30,  
187   "name": "40-Mile Air",  
188   "type": "airline"  
189 }  
190 > {  
191   "callsign": "IPB",  
192   "country": "India",  
193   "iata": "A1F",  
194   "icao": "IPB",  
195   "id": 31,  
196   "name": "40-Mile Air",  
197   "type": "airline"  
198 }  
199 > {  
200   "callsign": "IPB",  
201   "country": "India",  
202   "iata": "A1F",  
203   "icao": "IPB",  
204   "id": 32,  
205   "name": "40-Mile Air",  
206   "type": "airline"  
207 }  
208 > {  
209   "callsign": "IPB",  
210   "country": "India",  
211   "iata": "A1F",  
212   "icao": "IPB",  
213   "id": 33,  
214   "name": "40-Mile Air",  
215   "type": "airline"  
216 }  
217 > {  
218   "callsign": "IPB",  
219   "country": "India",  
220   "iata": "A1F",  
221   "icao": "IPB",  
222   "id": 34,  
223   "name": "40-Mile Air",  
224   "type": "airline"  
225 }  
226 > {  
227   "callsign": "IPB",  
228   "country": "India",  
229   "iata": "A1F",  
230   "icao": "IPB",  
231   "id": 35,  
232   "name": "40-Mile Air",  
233   "type": "airline"  
234 }  
235 > {  
236   "callsign": "IPB",  
237   "country": "India",  
238   "iata": "A1F",  
239   "icao": "IPB",  
240   "id": 36,  
241   "name": "40-Mile Air",  
242   "type": "airline"  
243 }  
244 > {  
245   "callsign": "IPB",  
246   "country": "India",  
247   "iata": "A1F",  
248   "icao": "IPB",  
249   "id": 37,  
250   "name": "40-Mile Air",  
251   "type": "airline"  
252 }  
253 > {  
254   "callsign": "IPB",  
255   "country": "India",  
256   "iata": "A1F",  
257   "icao": "IPB",  
258   "id": 38,  
259   "name": "40-Mile Air",  
260   "type": "airline"  
261 }  
262 > {  
263   "callsign": "IPB",  
264   "country": "India",  
265   "iata": "A1F",  
266   "icao": "IPB",  
267   "id": 39,  
268   "name": "40-Mile Air",  
269   "type": "airline"  
270 }  
271 > {  
272   "callsign": "IPB",  
273   "country": "India",  
274   "iata": "A1F",  
275   "icao": "IPB",  
276   "id": 40,  
277   "name": "40-Mile Air",  
278   "type": "airline"  
279 }  
280 > {  
281   "callsign": "IPB",  
282   "country": "India",  
283   "iata": "A1F",  
284   "icao": "IPB",  
285   "id": 41,  
286   "name": "40-Mile Air",  
287   "type": "airline"  
288 }  
289 > {  
290   "callsign": "IPB",  
291   "country": "India",  
292   "iata": "A1F",  
293   "icao": "IPB",  
294   "id": 42,  
295   "name": "40-Mile Air",  
296   "type": "airline"  
297 }  
298 > {  
299   "callsign": "IPB",  
300   "country": "India",  
301   "iata": "A1F",  
302   "icao": "IPB",  
303   "id": 43,  
304   "name": "40-Mile Air",  
305   "type": "airline"  
306 }  
307 > {  
308   "callsign": "IPB",  
309   "country": "India",  
310   "iata": "A1F",  
311   "icao": "IPB",  
312   "id": 44,  
313   "name": "40-Mile Air",  
314   "type": "airline"  
315 }  
316 > {  
317   "callsign": "IPB",  
318   "country": "India",  
319   "iata": "A1F",  
320   "icao": "IPB",  
321   "id": 45,  
322   "name": "40-Mile Air",  
323   "type": "airline"  
324 }  
325 > {  
326   "callsign": "IPB",  
327   "country": "India",  
328   "iata": "A1F",  
329   "icao": "IPB",  
330   "id": 46,  
331   "name": "40-Mile Air",  
332   "type": "airline"  
333 }  
334 > {  
335   "callsign": "IPB",  
336   "country": "India",  
337   "iata": "A1F",  
338   "icao": "IPB",  
339   "id": 47,  
340   "name": "40-Mile Air",  
341   "type": "airline"  
342 }  
343 > {  
344   "callsign": "IPB",  
345   "country": "India",  
346   "iata": "A1F",  
347   "icao": "IPB",  
348   "id": 48,  
349   "name": "40-Mile Air",  
350   "type": "airline"  
351 }  
352 > {  
353   "callsign": "IPB",  
354   "country": "India",  
355   "iata": "A1F",  
356   "icao": "IPB",  
357   "id": 49,  
358   "name": "40-Mile Air",  
359   "type": "airline"  
360 }  
361 > {  
362   "callsign": "IPB",  
363   "country": "India",  
364   "iata": "A1F",  
365   "icao": "IPB",  
366   "id": 50,  
367   "name": "40-Mile Air",  
368   "type": "airline"  
369 }  
370 > {  
371   "callsign": "IPB",  
372   "country": "India",  
373   "iata": "A1F",  
374   "icao": "IPB",  
375   "id": 51,  
376   "name": "40-Mile Air",  
377   "type": "airline"  
378 }  
379 > {  
380   "callsign": "IPB",  
381   "country": "India",  
382   "iata": "A1F",  
383   "icao": "IPB",  
384   "id": 52,  
385   "name": "40-Mile Air",  
386   "type": "airline"  
387 }  
388 > {  
389   "callsign": "IPB",  
390   "country": "India",  
391   "iata": "A1F",  
392   "icao": "IPB",  
393   "id": 53,  
394   "name": "40-Mile Air",  
395   "type": "airline"  
396 }  
397 > {  
398   "callsign": "IPB",  
399   "country": "India",  
400   "iata": "A1F",  
401   "icao": "IPB",  
402   "id": 54,  
403   "name": "40-Mile Air",  
404   "type": "airline"  
405 }  
406 > {  
407   "callsign": "IPB",  
408   "country": "India",  
409   "iata": "A1F",  
410   "icao": "IPB",  
411   "id": 55,  
412   "name": "40-Mile Air",  
413   "type": "airline"  
414 }  
415 > {  
416   "callsign": "IPB",  
417   "country": "India",  
418   "iata": "A1F",  
419   "icao": "IPB",  
420   "id": 56,  
421   "name": "40-Mile Air",  
422   "type": "airline"  
423 }  
424 > {  
425   "callsign": "IPB",  
426   "country": "India",  
427   "iata": "A1F",  
428   "icao": "IPB",  
429   "id": 57,  
430   "name": "40-Mile Air",  
431   "type": "airline"  
432 }  
433 > {  
434   "callsign": "IPB",  
435   "country": "India",  
436   "iata": "A1F",  
437   "icao": "IPB",  
438   "id": 58,  
439   "name": "40-Mile Air",  
440   "type": "airline"  
441 }  
442 > {  
443   "callsign": "IPB",  
444   "country": "India",  
445   "iata": "A1F",  
446   "icao": "IPB",  
447   "id": 59,  
448   "name": "40-Mile Air",  
449   "type": "airline"  
450 }  
451 > {  
452   "callsign": "IPB",  
453   "country": "India",  
454   "iata": "A1F",  
455   "icao": "IPB",  
456   "id": 60,  
457   "name": "40-Mile Air",  
458   "type": "airline"  
459 }  
460 > {  
461   "callsign": "IPB",  
462   "country": "India",  
463   "iata": "A1F",  
464   "icao": "IPB",  
465   "id": 61,  
466   "name": "40-Mile Air",  
467   "type": "airline"  
468 }  
469 > {  
470   "callsign": "IPB",  
471   "country": "India",  
472   "iata": "A1F",  
473   "icao": "IPB",  
474   "id": 62,  
475   "name": "40-Mile Air",  
476   "type": "airline"  
477 }  
478 > {  
479   "callsign": "IPB",  
480   "country": "India",  
481   "iata": "A1F",  
482   "icao": "IPB",  
483   "id": 63,  
484   "name": "40-Mile Air",  
485   "type": "airline"  
486 }  
487 > {  
488   "callsign": "IPB",  
489   "country": "India",  
490   "iata": "A1F",  
491   "icao": "IPB",  
492   "id": 64,  
493   "name": "40-Mile Air",  
494   "type": "airline"  
495 }  
496 > {  
497   "callsign": "IPB",  
498   "country": "India",  
499   "iata": "A1F",  
500   "icao": "IPB",  
501   "id": 65,  
502   "name": "40-Mile Air",  
503   "type": "airline"  
504 }  
505 > {  
506   "callsign": "IPB",  
507   "country": "India",  
508   "iata": "A1F",  
509   "icao": "IPB",  
510   "id": 66,  
511   "name": "40-Mile Air",  
512   "type": "airline"  
513 }  
514 > {  
515   "callsign": "IPB",  
516   "country": "India",  
517   "iata": "A1F",  
518   "icao": "IPB",  
519   "id": 67,  
520   "name": "40-Mile Air",  
521   "type": "airline"  
522 }  
523 > {  
524   "callsign": "IPB",  
525   "country": "India",  
526   "iata": "A1F",  
527   "icao": "IPB",  
528   "id": 68,  
529   "name": "40-Mile Air",  
530   "type": "airline"  
531 }  
532 > {  
533   "callsign": "IPB",  
534   "country": "India",  
535   "iata": "A1F",  
536   "icao": "IPB",  
537   "id": 69,  
538   "name": "40-Mile Air",  
539   "type": "airline"  
540 }  
541 > {  
542   "callsign": "IPB",  
543   "country": "India",  
544   "iata": "A1F",  
545   "icao": "IPB",  
546   "id": 70,  
547   "name": "40-Mile Air",  
548   "type": "airline"  
549 }  
550 > {  
551   "callsign": "IPB",  
552   "country": "India",  
553   "iata": "A1F",  
554   "icao": "IPB",  
555   "id": 71,  
556   "name": "40-Mile Air",  
557   "type": "airline"  
558 }  
559 > {  
560   "callsign": "IPB",  
561   "country": "India",  
562   "iata": "A1F",  
563   "icao": "IPB",  
564   "id": 72,  
565   "name": "40-Mile Air",  
566   "type": "airline"  
567 }  
568 > {  
569   "callsign": "IPB",  
570   "country": "India",  
571   "iata": "A1F",  
572   "icao": "IPB",  
573   "id": 73,  
574   "name": "40-Mile Air",  
575   "type": "airline"  
576 }  
577 > {  
578   "callsign": "IPB",  
579   "country": "India",  
580   "iata": "A1F",  
581   "icao": "IPB",  
582   "id": 74,  
583   "name": "40-Mile Air",  
584   "type": "airline"  
585 }  
586 > {  
587   "callsign": "IPB",  
588   "country": "India",  
589   "iata": "A1F",  
590   "icao": "IPB",  
591   "id": 75,  
592   "name": "40-Mile Air",  
593   "type": "airline"  
594 }  
595 > {  
596   "callsign": "IPB",  
597   "country": "India",  
598   "iata": "A1F",  
599   "icao": "IPB",  
600   "id": 76,  
601   "name": "40-Mile Air",  
602   "type": "airline"  
603 }  
604 > {  
605   "callsign": "IPB",  
606   "country": "India",  
607   "iata": "A1F",  
608   "icao": "IPB",  
609   "id": 77,  
610   "name": "40-Mile Air",  
611   "type": "airline"  
612 }  
613 > {  
614   "callsign": "IPB",  
615   "country": "India",  
616   "iata": "A1F",  
617   "icao": "IPB",  
618   "id": 78,  
619   "name": "40-Mile Air",  
620   "type": "airline"  
621 }  
622 > {  
623   "callsign": "IPB",  
624   "country": "India",  
625   "iata": "A1F",  
626   "icao": "IPB",  
627   "id": 79,  
628   "name": "40-Mile Air",  
629   "type": "airline"  
630 }  
631 > {  
632   "callsign": "IPB",  
633   "country": "India",  
634   "iata": "A1F",  
635   "icao": "IPB",  
636   "id": 80,  
637   "name": "40-Mile Air",  
638   "type": "airline"  
639 }  
640 > {  
641   "callsign": "IPB",  
642   "country": "India",  
643   "iata": "A1F",  
644   "icao": "IPB",  
645   "id": 81,  
646   "name": "40-Mile Air",  
647   "type": "airline"  
648 }  
649 > {  
650   "callsign": "IPB",  
651   "country": "India",  
652   "iata": "A1F",  
653   "icao": "IPB",  
654   "id": 82,  
655   "name": "40-Mile Air",  
656   "type": "airline"  
657 }  
658 > {  
659   "callsign": "IPB",  
660   "country": "India",  
661   "iata": "A1F",  
662   "icao": "IPB",  
663   "id": 83,  
664   "name": "40-Mile Air",  
665   "type": "airline"  
666 }  
667 > {  
668   "callsign": "IPB",  
669   "country": "India",  
670   "iata": "A1F",  
671   "icao": "IPB",  
672   "id": 84,  
673   "name": "40-Mile Air",  
674   "type": "airline"  
675 }  
676 > {  
677   "callsign": "IPB",  
678   "country": "India",  
679   "iata": "A1F",  
680   "icao": "IPB",  
681   "id": 85,  
682   "name": "40-Mile Air",  
683   "type": "airline"  
684 }  
685 > {  
686   "callsign": "IPB",  
687   "country": "India",  
688   "iata": "A1F",  
689   "icao": "IPB",  
690   "id": 86,  
691   "name": "40-Mile Air",  
692   "type": "airline"  
693 }  
694 > {  
695   "callsign": "IPB",  
696   "country": "India",  
697   "iata": "A1F",  
698   "icao": "IPB",  
699   "id": 87,  
700   "name": "40-Mile Air",  
701   "type": "airline"  
702 }  
703 > {  
704   "callsign": "IPB",  
705   "country": "India",  
706   "iata": "A1F",  
707   "icao": "IPB",  
708   "id": 88,  
709   "name": "40-Mile Air",  
710   "type": "airline"  
711 }  
712 > {  
713   "callsign": "IPB",  
714   "country": "India",  
715   "iata": "A1F",  
716   "icao": "IPB",  
717   "id": 89,  
718   "name": "40-Mile Air",  
719   "type": "airline"  
720 }  
721 > {  
722   "callsign": "IPB",  
723   "country": "India",  
724   "iata": "A1F",  
725   "icao": "IPB",  
726   "id": 90,  
727   "name": "40-Mile Air",  
728   "type": "airline"  
729 }  
730 > {  
731   "callsign": "IPB",  
732   "country": "India",  
733   "iata": "A1F",  
734   "icao": "IPB",  
735   "id": 91,  
736   "name": "40-Mile Air",  
737   "type": "airline"  
738 }  
739 > {  
740   "callsign": "IPB",  
741   "country": "India",  
742   "iata": "A1F",  
743   "icao": "IPB",  
744   "id": 92,  
745   "name": "40-Mile Air",  
746   "type": "airline"  
747 }  
748 > {  
749   "callsign": "IPB",  
750   "country": "India",  
751   "iata": "A1F",  
752   "icao": "IPB",  
753   "id": 93,  
754   "name": "40-Mile Air",  
755   "type": "airline"  
756 }  
757 > {  
758   "callsign": "IPB",  
759   "country": "India",  
760   "iata": "A1F",  
761   "icao": "IPB",  
762   "id": 94,  
763   "name": "40-Mile Air",  
764   "type": "airline"  
765 }  
766 > {  
767   "callsign": "IPB",  
768   "country": "India",  
769   "iata": "A1F",  
770   "icao": "IPB",  
771   "id": 95,  
772   "name": "40-Mile Air",  
773   "type": "airline"  
774 }  
775 > {  
776   "callsign": "IPB",  
777   "country": "India",  
778   "iata": "A1F",  
779   "icao": "IPB",  
780   "id": 96,  
781   "name": "40-Mile Air",  
782   "type": "airline"  
783 }  
784 > {  
785   "callsign": "IPB",  
786   "country": "India",  
787   "iata": "A1F",  
788   "icao": "IPB",  
789   "id": 97,  
790   "name": "40-Mile Air",  
791   "type": "airline"  
792 }  
793 > {  
794   "callsign": "IPB",  
795   "country": "India",  
796   "iata": "A1F",  
797   "icao": "IPB",  
798   "id": 98,  
799   "name": "40-Mile Air",  
800   "type": "airline"  
801 }  
802 > {  
803   "callsign": "IPB",  
804   "country": "India",  
805   "iata": "A1F",  
806   "icao": "IPB",  
807   "id": 99,  
808   "name": "40-Mile Air",  
809   "type": "airline"  
810 }  
811 > {  
812   "callsign": "IPB",  
813   "country": "India",  
814   "iata": "A1F",  
815   "icao": "IPB",  
816   "id": 100,  
817   "name": "40-Mile Air",  
818   "type": "airline"  
819 }
```

Revert the country on the Target Cluster.

Edit Document X

airline\_10 metadata

```
1 ↵ {
2   "callsign": "MILE-AIR",
3   "country": "United States",
4   "iata": "Q5",
5   "icao": "MLA",
6   "id": 10,
7   "name": "40-Mile Air",
8   "type": "airline"
9 }
```

And save it.

Verify the record on the source cluster.

```
1 <pre>{
2   "callsign": "MILE-AIR",
3   "country": "India",
4   "iata": "Q5",
5   "icao": "MLA",
6   "id": 10,
7   "name": "40-Mile Air",
8   "type": "airline"
9 }</pre>
```

Its still India. Why? Reason is we have configured a single directional replication from Tos.Couchbase → Tos.CouchbaseII.

Now let us configure a bidirectional i.e replication from CouchbaseII → Couchbase. Access the web console of TOS.CouchbaseII and configure XDCR as shown below:

XDCR → Add remote Cluster

**Add Remote Cluster**

X

Cluster Name  
TOS.Couchbase

IP/Hostname ⓘ  
tos.hp.com

Username for Remote Cluster  
Administrator

Password  
.....|

Enable Secure Connection ⓘ

Cancel Save

Save and configure Replication

Add replication:

The screenshot shows the 'Add Replication' dialog box. It has the following fields:

- Replicate From Bucket:** travel-sample
- Remote Cluster:** TOS.Couchbase
- Remote Bucket:** travel-sample
- XDCR Protocol:** Version 2
- Enable advanced filtering:**  (highlighted with a red underline)
- Filter Expression:** airline (highlighted with a red underline)
- Test Key:** 1 airline\_SFO (highlighted with a red underline)
- Match:** (green button)
- Show Advanced Settings:** (link)

**Click Enable advanced filtering**

Advanced Filtering can be enabled by checking the Enabled advanced filtering checkbox. The UI expands to reveal the following field:

Enable advanced filtering

Filter Expression ⓘ

Test Key

No Match

1

One or more regular expressions, to be used as filters, can be entered into the Filter Expression field. The expression is matched against document keys within the source bucket. (Note that no match is attempted with document values.) If a match is successful, that document is replicated. Documents whose keys do not provide a match are not replicated.

Optionally, document keys can be entered into a Test Key field: hitting return produces successive fields, so any number of keys can be tested. If a match is successful, the orange No Match button is displayed in green, and signifies that a Match has been made:

Enable advanced filtering

Filter Expression ⓘ

  
airline

Test Key

Match

1 airline\_SFO

Or using 7.0 UI

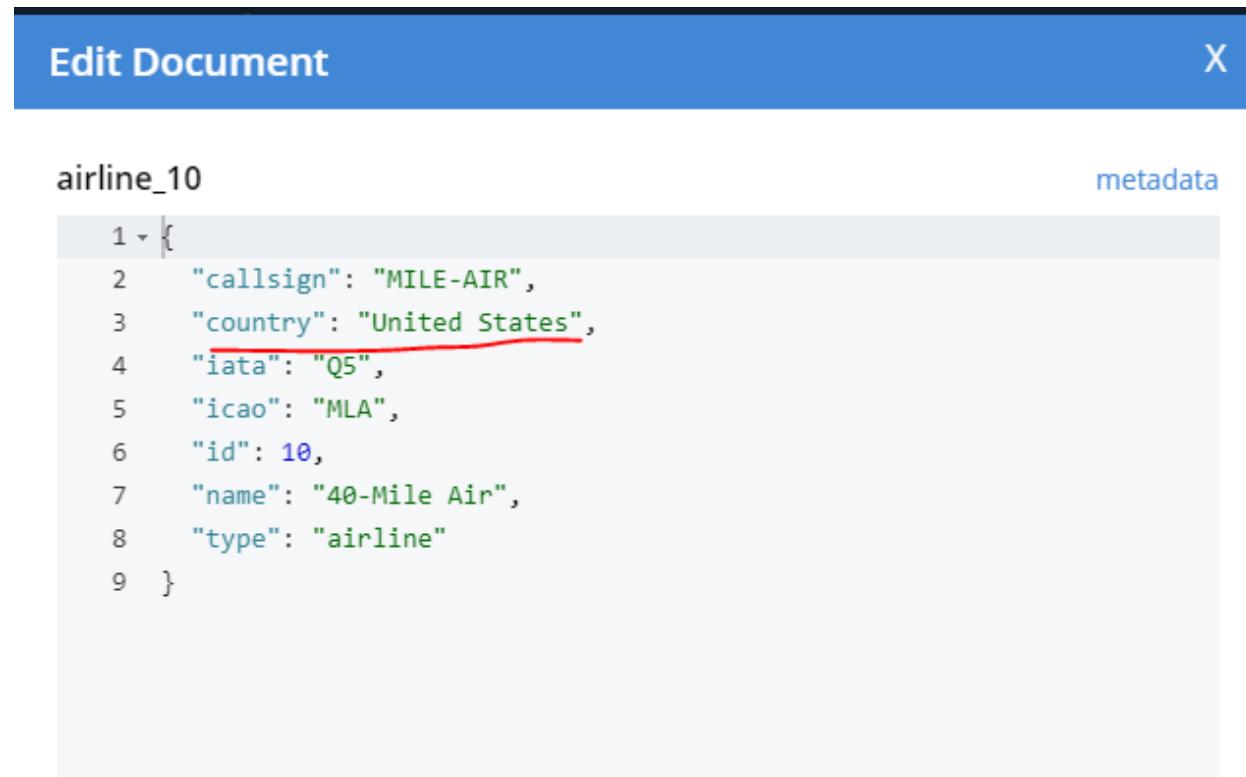
To replicate only those documents whose body contains *France* as the value of *country*, enter the expression '*REGEXP\_CONTAINS(country, "France")*', in the **Filter Expression** field:

The screenshot shows the 'Filter Expression' configuration for a replication task. The expression entered is 'REGEXP\_CONTAINS(Learning,'over')'. A tooltip explains it as 'Learning : field contains over'. Below this, there's a 'Test Filter Expression' section with input fields for 'Learning' and 'over', and a 'Test Filter' button which shows a green checkmark and the word 'match'. There are also two radio buttons: one for 'Save filter & restart replication' (selected) and one for 'Save & continue replicating'.

After saving it should be as shown below:

The screenshot shows the 'XDCR Replications' page. On the left, a sidebar lists 'Remote Clusters', 'Ongoing Replications', and other tabs like 'Logs' and 'Metrics'. The main area shows a table for 'Remote Clusters' with one entry: 'TOS.Couchbase' connected to 'tos.hp.com:8091'. Below this, the 'Ongoing Replications' table shows a single replication task for the 'travel-sample' bucket from 'this cluster' to 'bucket "travel-sample" on cluster "TOS.Couchbase"'. The status is 'Replicating' (highlighted with a red circle). The table has columns: bucket, protocol, from, to, filtered, status, and when.

Now you can verify the record on the TOS.Couchbase cluster.



Edit Document X

airline\_10 metadata

```
1 {  
2   "callsign": "MILE-AIR",  
3   "country": "United States",  
4   "iata": "Q5",  
5   "icao": "MLA",  
6   "id": 10,  
7   "name": "40-Mile Air",  
8   "type": "airline"  
9 }
```

As shown above the country is not United States. Refresh the console if required.

Access Statistics for the Source Bucket i.e TOS.Cluster

Proceeds as follows:

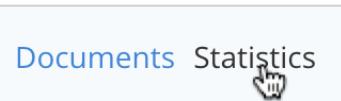
1. Ascertain the name of the XDCR source bucket to be examined.
2. In the left-hand navigation panel for the source cluster, left-click on the Buckets tab:



This displays the Buckets screen. A single bucket is displayed, travel-sample which is the source for replication.

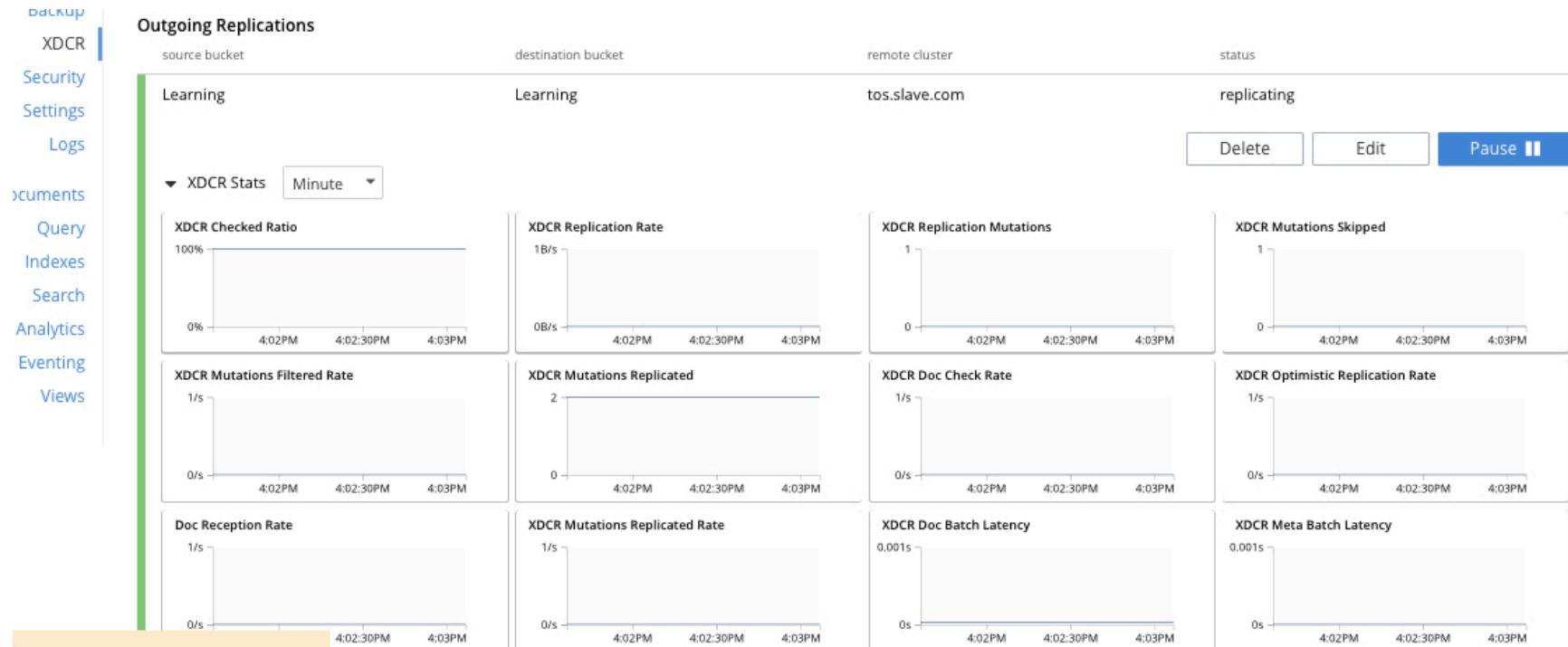
name ▾	items	resident	ops/sec	RAM used/quota	disk used	
travel-sample	3,111	100%	0	16.7MB / 100MB	28.6MB	<a href="#">Documents</a> <a href="#">Statistics</a>

3. Left-click on the Statistics tab, which is located at the right-hand side of the row for the bucket:

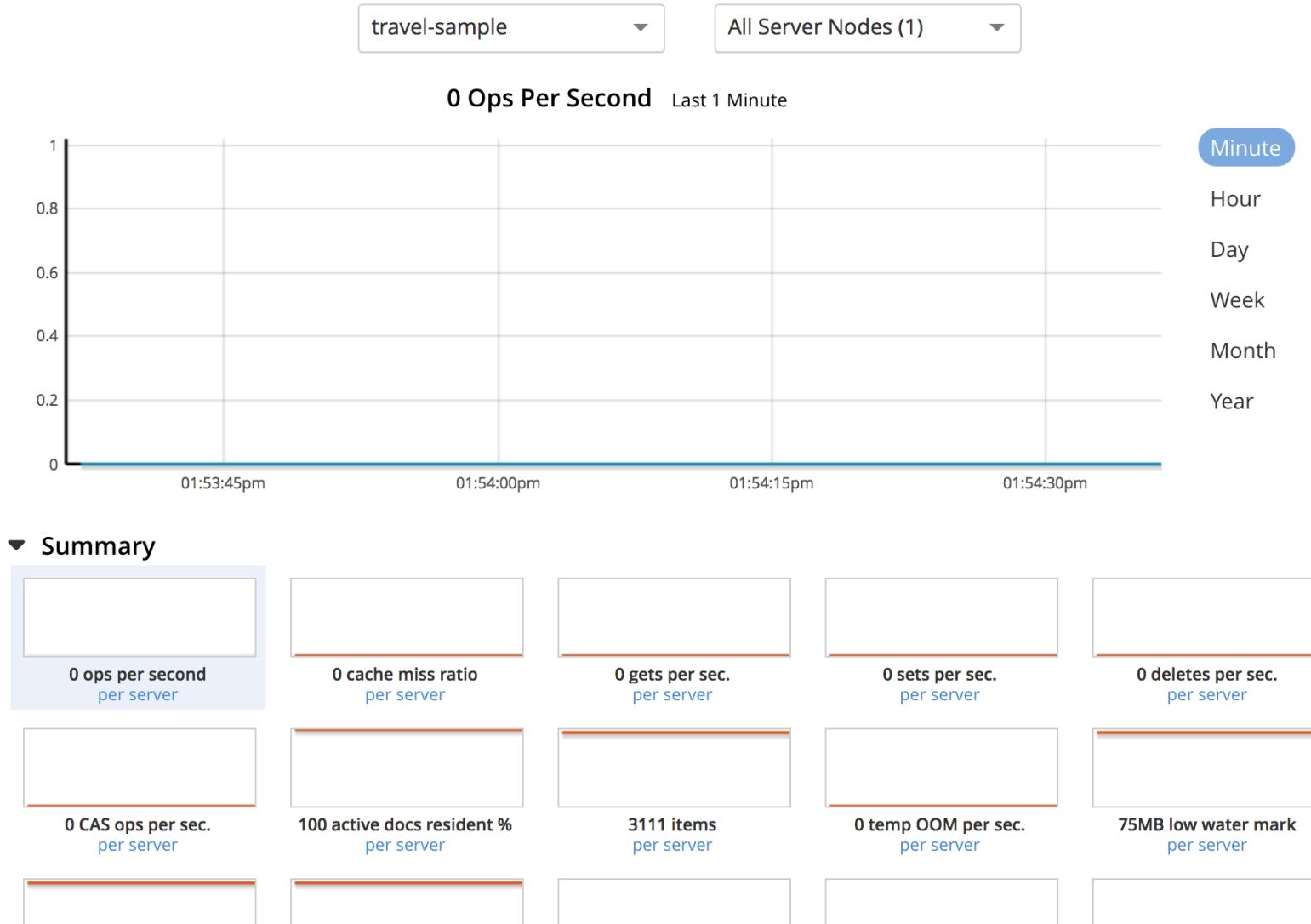


Or in web UI- 7.0

## XDCR -> Outgoing Replications:



The Statistics screen is now displayed:

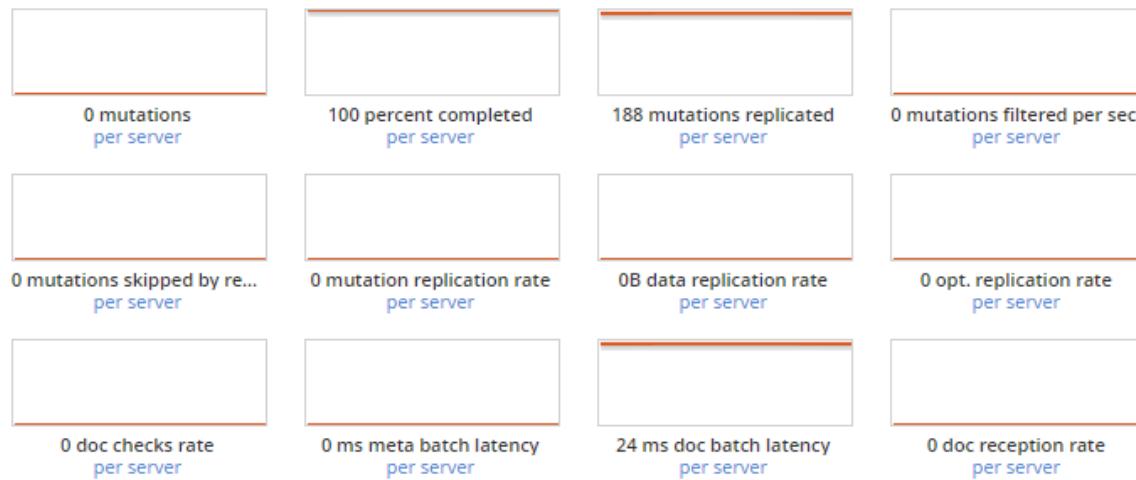


4. Scroll down the screen, until the XDCR statistics headers appear:

- ▼ Outbound XDCR Operations to bucket "travel-sample" on remote cluster "TOS.Couchbase"

5. Left-click on the arrow for Outbound XDCR operations. The following graphical display is provided:

- ▼ Outbound XDCR Operations to bucket "travel-sample" on remote cluster "TOS.Couchbase"



## Delete an XDCR Replication with the UI using TOS.Couchbase

Proceed as follows:

1. Access Couchbase Web Console. Left-click on the XDCR tab, in the left-hand navigation menu.



This displays the XDCR Replications screen. The lower part of the main panel, entitled Remote Clusters, features and Ongoing Replications panel that currently has the following appearance:

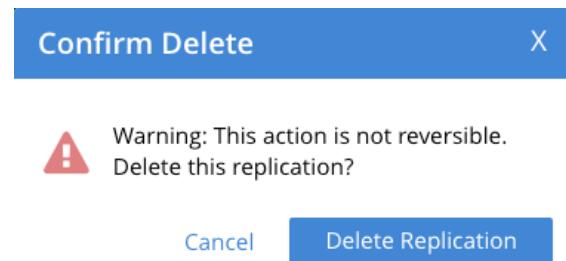
Ongoing Replications							Add Replication
bucket	protocol	from	to	filtered	status	when	
travel-sample	Version 2	this cluster	bucket "travel-sample" on cluster "10.142.180.102"	No	Replicating		<a href="#">Delete</a> <a href="#">Edit</a>

This features information on a single, currently defined replication. In the status column, this replication is shown to be Replicating.

2. To delete the replication, left-click on the Delete tab, which appears near the right-hand side of the row:



The following confirmation dialog is now displayed:



Left-click on Delete Replication, to confirm. The Ongoing Replications panel now reappears, showing no replications:

Ongoing Replications							Add Replication
bucket	protocol	from	to	filtered	status	when	
There are no replications currently in progress.							

The replication has now been deleted

Do the same steps using the TOS.Couchbase11 console.

Ongoing Replications							Add Replication
bucket	protocol	from	to	filtered	status	when	
travel-sample	Version 2	this cluster	bucket "travel-sample" on cluster "TOS.Couchbase"	Yes	Replicating 		<a href="#">Delete</a> <a href="#">Edit</a>

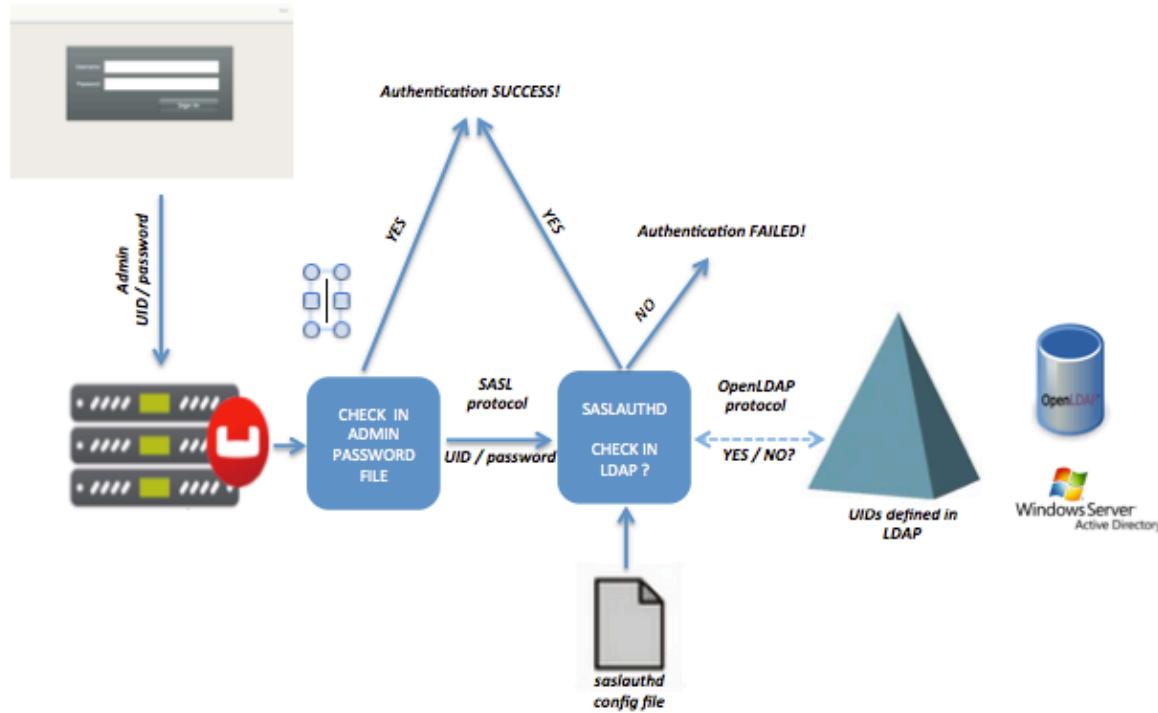
----- Lab Ends Here -----

## 19. LDAP Authentication & Security Roles – 90 Minutes(DP)

LDAP authentication for Couchbase administrators involves setting up LDAP administrators on the LDAP server, mapping their user IDs using the Couchbase Web Console and configuring the saslauthd agent.

**Attention: Remote authentication with LDAP is available in the Enterprise Edition of the Couchbase Server only for the Linux platform. It is not available for Windows or Mac OS.**

**Attention:** Mixed version cluster deployments do not support LDAP authentication: upgrade all nodes to the latest Couchbase Server release to use LDAP authentication.



## LDAP Installation

### LDAP server software

The LDAP server software is downloaded and installed separately on the LDAP server. This document explains how to install openssl and configured to work with Couchbase Server.

Perform these tasks on the LDAP server:

- Create users.
- Set up user passwords.

These tasks are performed using the Couchbase Web Console:

- Mapping users in LDAP to full administrators or read-only administrators in Couchbase.
- Validating LDAP credentials.

## **saslauthd**

The saslauthd process handles authentication requests on behalf of Couchbase Server.

To use LDAP authentication, you need to configure saslauthd properly using the steps explained below.

## Configuring LDAP on the Server

The Lightweight Directory Access Protocol (LDAP) is a public standard that facilitates distributed directories (such as network user privilege information) over the Internet Protocol (IP).

Install the **openldap**, **openldap-servers**, and **openldap-clients** RPMs.

Refer Annexure – LDAP Installation (use Windows unless specified.)

Couchbase connects to LDAP through the saslauthd library.

```
#yum install openldap-clients
```

```
=====  
Installing:  
  openldap-clients      x86_64      2.4.23-31.el6      rhel6-Server      165 k  
  
Transaction Summary  
=====  
Install       1 Package(s)  
  
Total download size: 165 k  
Installed size: 609 k  
Is this ok [y/N]: y  
Downloading Packages:  
Running rpm_check_debug  
Running Transaction Test  
Transaction Test Succeeded  
Running Transaction  
  Installing : openldap-clients-2.4.23-31.el6.x86_64          1/1  
  Verifying   : openldap-clients-2.4.23-31.el6.x86_64          1/1  
  
Installed:  
  openldap-clients.x86_64 0:2.4.23-31.el6  
  
Complete!  
[root@localhost ~]#
```

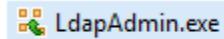
Let us create two user using LDAP Client

Perform these tasks on the LDAP server from your windows laptop.

Let us create two users which will be integrated with Couchbase cluster and Set up user passwords

Name

Double click LdapAdmin.exe on your desktop window



You can download from the url, <http://www.ldapadmin.org/download/ldapadmin.html> if not present in your desktop.

## Downloads

LdapAdmin 1.8.3

[LdapAdminExe-w32-1.8.3.zip](#)

[LdapAdminExe-w64-1.8.3.zip](#)

Click : Start --> Connect --> New Connection

Supply the following details: Host will be the IP of the server where openLdap is running. Unselect the Anonymous connection.

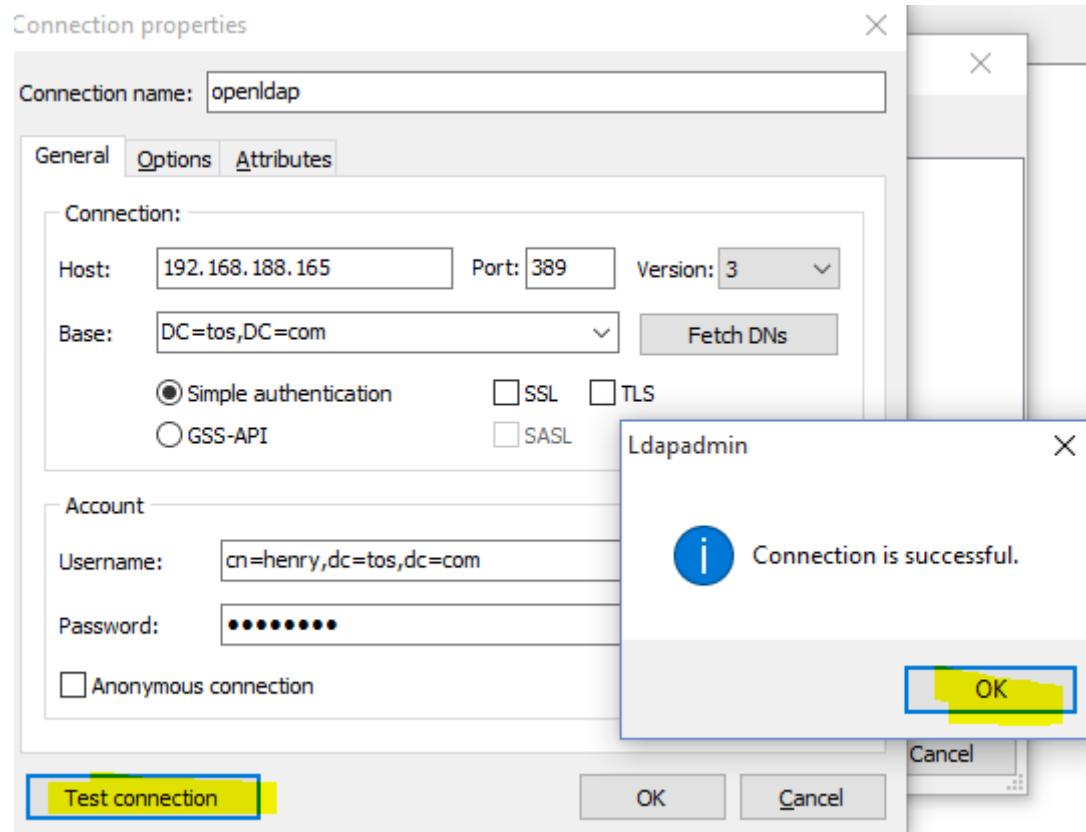
Connection Name : openldap

Host: 192.168.188.165 [hostname in windows use localhost]

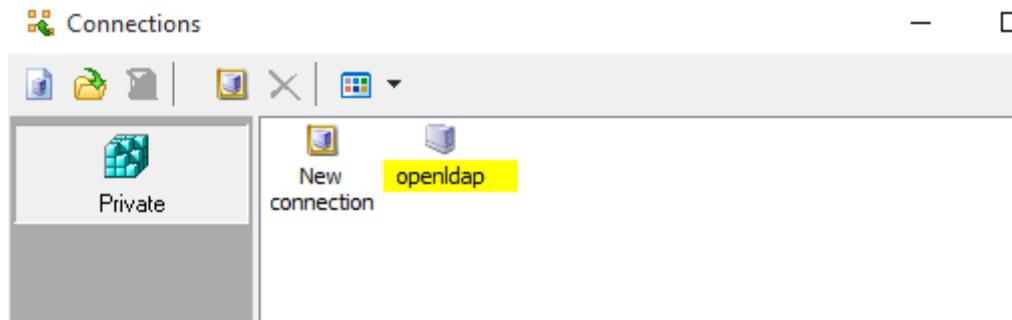
Base:DC=tos,DC=com

Username: cn=henry,dc=tos,dc=com

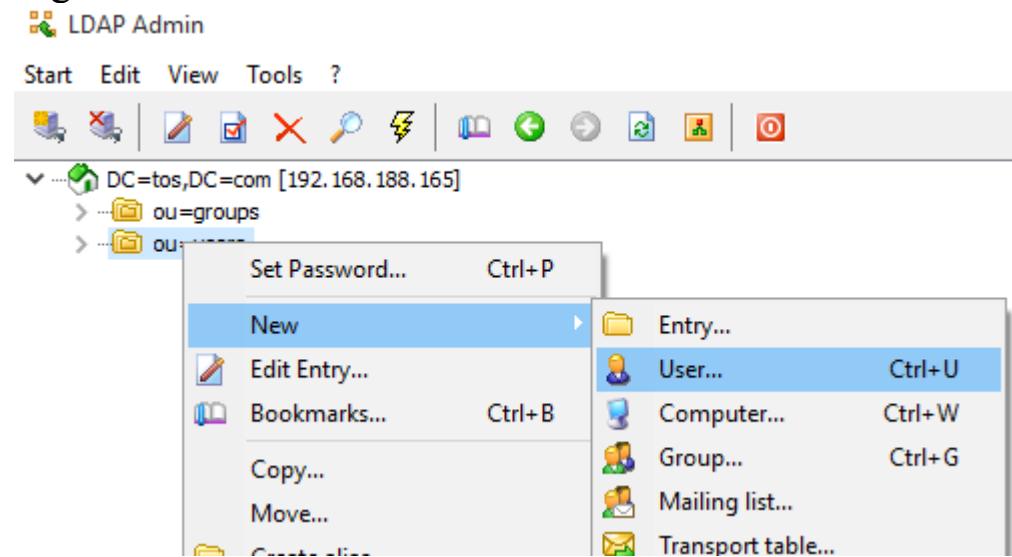
Password: secret



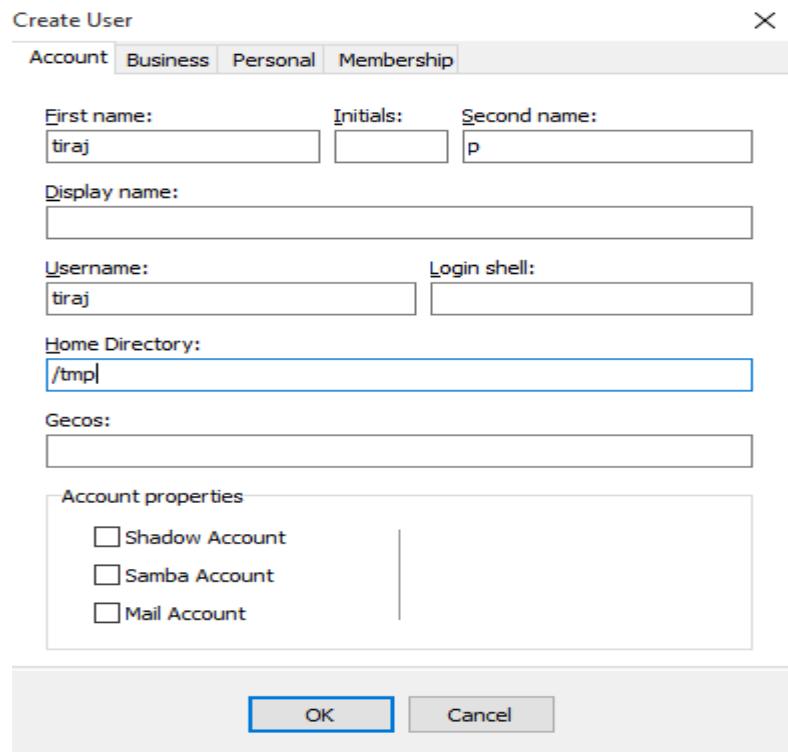
Click Test Connection --> It's should show that connection is successful. Click Ok.  
You need to create two users as shown below:  
Double click on openldap.



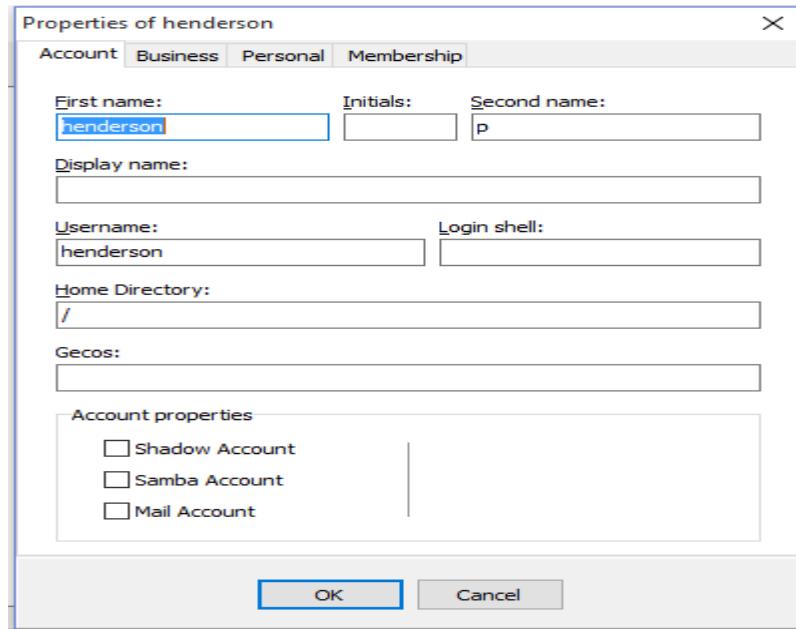
Right click on the node users --> Create User as shown below.



Create two users with the following details  
tiraj



henderson:



You can set password as: tiraj --> tiraj , henderson --> tiraj

The screenshot shows an LDAP browser interface with the URL 'DC=tos,DC=com [192.168.188.165]'. On the left, a tree view shows 'ou=groups' and 'ou=users'. Under 'ou=users', there are two entries: 'uid=henderson' and 'uid=tiraj'. A 'Set Password' dialog box is overlaid on the interface. It has fields for 'New password:' containing '\*\*\*\*\*', 'Confirm password:' containing '\*\*\*\*\*', and 'Encryption method:' set to 'SHA1'. It also has 'OK' and 'Cancel' buttons.

Execute from the LDAP installation folder path : D:\MyExperiment\OpenLDAP\ClientTools>

Let us verify the users now . Note password should be secret

```
#ldapsearch -x -W -D "cn=henry,dc=tos,dc=com" -b "dc=tos,dc=com" "(objectclass=*)"  
# numEntries: 3  
[root@localhost couchbase]# ldapsearch -x -W -D "cn=henry,dc=tos,dc=com" -b "dc=tos,dc=com" "(objectclass=*)" "
```

You should be able to determine an entry as shown below.

dn: uid=tiraj,ou=users,dc=tos,dc=com

```
# tiraj, users, tos.com  
dn: uid=tiraj,ou=users,dc=tos,dc=com  
objectClass: posixAccount  
objectClass: top  
objectClass: inetOrgPerson  
gidNumber: 0  
givenName: tiraj  
sn: p  
uid: tiraj  
homeDirectory: /tmp  
cn: tiraj  
uidNumber: 1775  
userPassword:: e1NIQX1oW1ZyZysw5jJBdm9JeGt1OWRBdEQyNFVSTDg9
```

Setting up saslauthd – It should be executed on the Couchbase cluster. i.e our centos server.

### Install saslauthd

**Install the saslauthd package on your operating system**

#yum install cyrus-sasl

```

Total                                         157 kB/s | 243 kB  00:01
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Updating   : cyrus-sasl-lib-2.1.26-21.el7.x86_64          1/3
  Installing : cyrus-sasl-2.1.26-21.el7.x86_64            2/3
  Cleanup    : cyrus-sasl-lib-2.1.26-20.el7_2.x86_64        3/3
  Verifying   : cyrus-sasl-lib-2.1.26-21.el7.x86_64          1/3
  Verifying   : cyrus-sasl-2.1.26-21.el7.x86_64            2/3
  Verifying   : cyrus-sasl-lib-2.1.26-20.el7_2.x86_64        3/3

Installed:
  cyrus-sasl.x86_64 0:2.1.26-21.el7

Dependency Updated:
  cyrus-sasl-lib.x86_64 0:2.1.26-21.el7

Complete!
[root@tos ~]# yum install cyrus-sasl

```

saslauthd is a daemon process that handles plaintext authentication requests on behalf of the SASL library.

In LDAP authentication, the saslauthd process handles authentication requests on behalf of Couchbase Server while the LDAP protocol is used to connect to the LDAP server.

**Important:** Remote authentication with the LDAP server requires proper configuration of the saslauthd agent, which must be installed and configured on each Couchbase Server node.

### Configuring saslauthd Library for LDAP

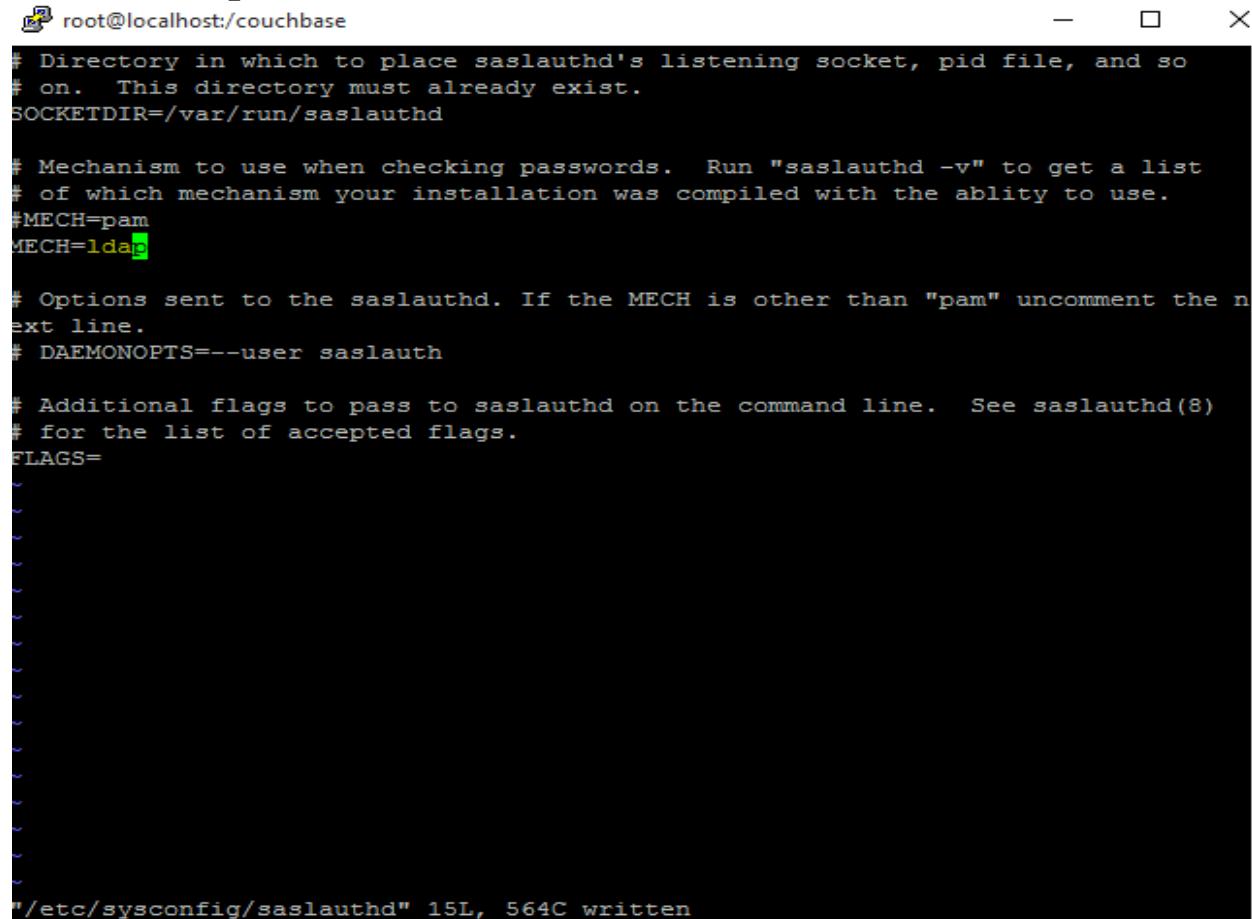
Depending on the system, the saslauthd file is configured as follows:

#### **Red Hat Enterprise Linux, CentOS, and Amazon Linux AMI**

# **vi /etc/sysconfig/saslauthd**

If you are using a system that configures saslauthd with the file /etc/sysconfig/saslauthd, such as Red Hat Enterprise Linux, CentOS, and Amazon Linux AMI, set the mechanism MECH to ldap:

## MECH=ldap



```
# Directory in which to place saslauthd's listening socket, pid file, and so
# on. This directory must already exist.
SOCKETDIR=/var/run/saslauthd

# Mechanism to use when checking passwords. Run "saslauthd -v" to get a list
# of which mechanism your installation was compiled with the ability to use.
#MECH=pam
MECH=ldap

# Options sent to the saslauthd. If the MECH is other than "pam" uncomment the n
ext line.
# DAEMONOPTS=--user saslauth

# Additional flags to pass to saslauthd on the command line. See saslauthd(8)
# for the list of accepted flags.
FLAGS=

"/etc/sysconfig/saslauthd" 15L, 564C written
```

Important: Change permissions for the saslauthd directory!

Don't forget to set the correct access permissions for the saslauthd directory:

```
#chmod 755 /var/run/saslauthd
#chmod 755 /etc/sysconfig/saslauthd
#chmod 755 /usr/sbin/saslauthd
```

```
[root@localhost couchbase]# chmod 755 /var/run/saslauthd
```

Or

```
[root@tos ~]# chmod 755 /etc/sysconfig/saslauthd  
[root@tos ~]# chmod 755 /usr/sbin/saslauthd
```

Configuring the saslauthd Configuration File

Skip in case the folder doesn't exist.

The default configuration file used to obtain the LDAP configuration parameters is located at /usr/local/etc/saslauthd.conf.

### **Step 1: Set up ldap\_servers (It should be the IP of the LDAP Server.)**

Specify URIs of the LDAP servers used for authentication, such as ldap://10.1.1.11/,ldap://10.1.1.12/. Multiple LDAP servers can be specified in the list, which is then tested to find out whether one of the servers is offline. If you install OpenLDAP on the local host machine, you can specify the value ldap://localhost:389.

### **Step 2: Set up ldap\_search\_base**

Specify the distinguished name to which the search is relative. The search includes the base or objects below.

It also includes Domain Components ( dc) such as in dc=tos and dc=com.

### **Step 3: Set up ldap\_filter**

Specify the search filter. The values for these configuration options correspond to the values specific to the test. For example, to filter on email specify ldap\_filter: (mail=%n).

ldap\_filter: (uid=%u)

Create a file and paste the following content. Replace the IP with that of your windows machine[hints: ipconfig]

```
Ethernet adapter Ethernet 5:

Connection-specific DNS Suffix . . .
Link-local IPv6 Address . . . . . : fe80::242c:e3a6:3053:94a4%21
IPv4 Address. . . . . : 10.10.20.1
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . :
```

```
#vi /etc/saslauthd.conf
```

```
ldap_servers: ldap://10.10.20.1
ldap_search_base:dc=tos,dc=com
ldap_filter: (uid=%u)
ldap_bind_dn: CN=henry,DC=tos,DC=com
ldap_password: secret
```

```
#cat /etc/sysconfig/saslauthd
```

```
[root@localhost etc]# cat /etc/sysconfig/saslauthd
# Directory in which to place saslauthd's listening socket, pid file, and so
# on. This directory must already exist.
SOCKETDIR=/var/run/saslauthd

# Mechanism to use when checking passwords. Run "saslauthd -v" to get a list
# of which mechanism your installation was compiled with the ability to use.
#MECH=pam
MECH=ldap

# Options sent to the saslauthd. If the MECH is other than "pam" uncomment the n
ext line.
#DAEMONOPTS="--user saslauth

# Additional flags to pass to saslauthd on the command line. See saslauthd(8)
# for the list of accepted flags.
#FLAGS="-O /etc/saslauthd.conf"
[root@localhost etc]#
```

Restart your saslauthd service

```
#service saslauthd restart
```

```
[root@localhost couchbase]# service saslauthd status
saslauthd (pid 4896) is running...
[root@localhost couchbase]# service saslauthd restart
Stopping saslauthd: [ OK ]
Starting saslauthd: [ OK ]
[root@localhost couchbase]#
```

Test saslauthd

If the connection is properly working, the user couchbase must have access to /var/run/saslauthd/mux (or the appropriate another folder for SUSE) in order to communicate to saslauthd

```
chkconfig saslauthd on
```

Verify the service is on with any of the following command depending on the OS.

**chkconfig --list saslauthd**

or

**systemctl list-unit-files | grep saslauthd**

```
[root@localhost etc]# chkconfig saslauthd on
[root@localhost etc]# chkconfig --list saslauthd
saslauthd      0:off    1:off    2:on     3:on     4:on     5:on     6:off
[root@localhost etc]# [REDACTED]

[root@tos ~]# systemctl list-unit-files | grep saslauthd
saslauthd.service                           enabled
[root@tos ~]# [REDACTED]
```

Test it!

**#testsaslauthd -u tiraj -p tiraj -f /var/run/saslauthd/mux**

```
[root@localhost etc]#
[root@localhost etc]# testsaslauthd -u tiraj -p tiraj -f /var/run/saslauthd/mux
0: OK "Success."
[root@localhost etc]# [REDACTED]
```

Notes: Logging /var/log/messages or /var/log/secure

## External Roles

The Full Administrator can configure other Couchbase administrators when LDAP authentication is enabled.

Since the Couchbase Web Console can only read the LDAP database (and cannot write to it), all external administrators must be created on the LDAP server. After the user IDs of these administrators have been defined, they can be mapped to Couchbase Server using the Couchbase Web Console.

Enabling LDAP Authentication ( -C → It's the Couchbase server IP)

```
# cd /opt/couchbase/bin
```

```
./couchbase-cli setting-ldap -c tos.hp.com --username Administrator --password admin123 --ldap-enabled 1
```

```
[root@tos ~]# cd /opt/couchbase/bin
[root@tos bin]# ./couchbase-cli setting-ldap -c tos.hp.com --username Administrator --password admin123 --ldap-enabled 1
SUCCESS: LDAP settings modified
[root@tos bin]# █
```

Or you can enable the users along with the above command. [Optional]

```
./couchbase-cli setting-ldap -c tos.hp.com --username Administrator --password admin123 --ldap-enabled 1 --ldap-admins tiraj,henderson
```

Authentication with LDAP is disabled by default, and if you chose not to enable it, you wouldn't be able to set up the external administrative roles.

## Add Administrators

Use : Web console → Security ( There will be a message – External Authentication is enabled as shown below)

The screenshot shows the Couchbase Web Console interface. On the left, there's a vertical sidebar with links: Dashboard, Servers, Buckets, XDCR, Security (which is selected and highlighted in orange), Settings, and Logs. The main content area has a blue header bar with the title "Security". Below the header, there are tabs: "Users" (selected), "Root", and "Session". On the right side of the header, there are "FILTER" and "ADD USER" buttons. The main content area has a table with columns: "username", "full name", "auth domain", and "password set". A message at the bottom of the table says, "You don't have any users to display yet. Use ADD USER to add a new user." Above this message, there's a note: "✓ external authentication is enabled".

Note: Add the following user following the below steps:

Click Add user

user ID : Henderson Pwd : Henderson123

Authentication Domain : External.

roles: **beer-sample** →

### Add New User

X

**Authentication Domain**

Couchbase  External

**Username**

Henderson

**Full Name (optional)**

Henderson

**Roles**

- ▶ Administration & Global Roles
- ▶ All Buckets (\*)
- ▼ beer-sample
  - Bucket Admin ✓
  - Application Access ✓
  - XDCR Inbound ✓
- ▶ Data Service
- ▶ Views
- ▶ Query and Index Services
- ▶ Search Service
- ▶ Analytics Service
- ▶ travel-sample

Cancel Add User

## 7.0 UI

Add New User X

Username: Henderson

Full Name (optional):

Password: .....

Verify Password: .....

Roles Groups

Administrative

Bucket

Bucket Admin ⓘ Learning ADD

Manage Scopes ⓘ

Application Access ⓘ Learning ADD

Data

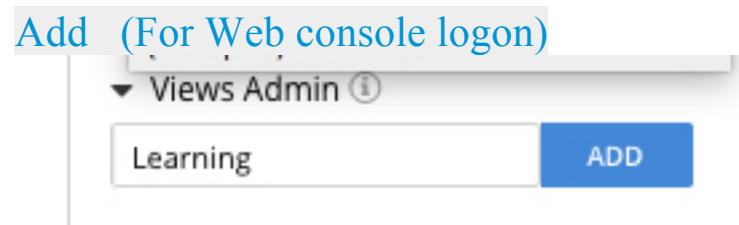
Views

Query & Index

Search

Analytics

Cancel Add User



Click Add User

Add another user as shown below:

User Id: Tiraj / tiraj123

role: Full Admin

### Add New User

X

Authentication Domain  
 Couchbase  External

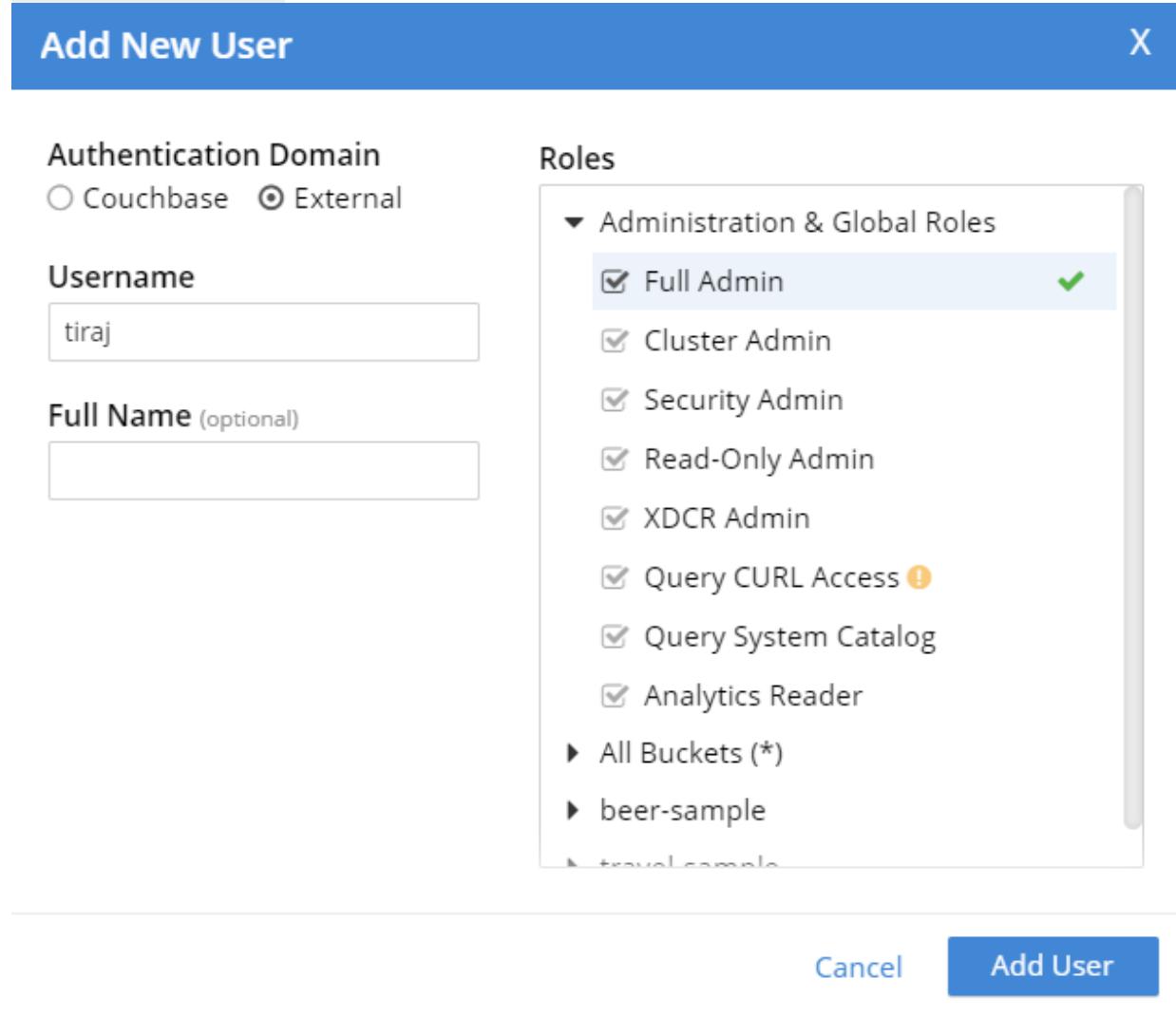
Username  
tiraj

Full Name (optional)

Roles

- ▼ Administration & Global Roles
  - Full Admin ✓
  - Cluster Admin
  - Security Admin
  - Read-Only Admin
  - XDCR Admin
  - Query CURL Access !
  - Query System Catalog
  - Analytics Reader
- ▶ All Buckets (\*)
- ▶ beer-sample

Cancel Add User



## Click Add User

At the end of the mention steps you should have administration right as follows:

username ▾	full name	roles	auth domain	password set
Henderson	Henderson	XDCR Inbound[beer-sample], Application Access[beer-sample], Bucket Admin[beer-sample]	External	
tiraj		Full Admin	External	

With LDAP authentication enabled, the Full Administrator can add administrators by setting up their roles and credentials.

### Test LDAP Settings

To test the LDAP settings:

Sign out of the Couchbase Web Console.

Restart the couchbase server.

```
#systemctl restart couchbase-server
```

Try to log in with the new administrative credentials.

If you enter the credentials of the Full Administrator, the screen will provide full access to all functions available through the Couchbase Web Console (see the LDAP Auth Setup screen above).

If you enter credentials of the Bucket Roles, a screen with the read-only view will become available. This screen doesn't allow the user to enable or disable LDAP, or to configure administrators.

When Log on as henderson:

name	items	resident	ops/sec	RAM used/quota	disk used
<a href="#">beer-sample</a>	7,303	100%	0	32MB / 200MB	53.7MB
<a href="#">travel-sample</a>	31,592	100%	0	75MB / 200MB	111MB

As highlighted above, the Henderson can only access the beer-sample documents. Click on the Documents link to verify the documents.

Bucket	Limit	Offset	Document ID	Where	Retrieve Docs
beer-sample	10	0	enter document ID or leave blank	e.g., 'meta().id = "some_id" and typ...	Retrieve Docs

10 Results for beer-sample, limit: 10, offset: 0

simple  spreadsheet

< Prev Batch | Next Batch >

<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	21st_amendment_brewery_cafe	{"address": ["563 Second Street"], "city": "San Francisco", "code": "94107", "country": "United States", "description": "The 21st Amendment Brewery offers a variety of award winning house made brews and Americ..."}  <input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	21st_amendment_brewery_cafe-21a_ipa	{"abv": 7.2, "brewery_id": "21st_amendment_brewery_cafe", "category": "North American Ale", "description": "Deep golden color. Citrus and piney hop aromas. Assertive malt backbone supporting the overwhelming..."}  <input checked="" type="checkbox"/>

When log on as tiraj:

The screenshot shows the Couchbase Dashboard interface. At the top, there are summary statistics: 2 active nodes, 0 failed-over nodes, 0 nodes pending rebalance, and 0 inactive nodes. Below this, a row of service status cards shows: Data (2 nodes), Index (2 nodes), Search (1 node), Query (1 node), Eventing (1 node), Analytics (1 node), and XDCR (1 remote cluster, 0 replications). The left sidebar contains navigation links for Dashboard, Servers, Buckets, XDCR, Security, Settings (selected), Logs, Documents, Query, Search, Analytics, Eventing, and Indexes. The main area displays two horizontal bar charts: 'Data Service Memory' (total quota 400 MB) and 'Data Service Disk' (usable free space 26.4 GB). Below these are two line charts: 'Buckets Operations Per Second' and 'Disk Fetches Per Second', both showing data over time from 10:30am to 11:15am.

Click on Buckets:

The screenshot shows the Couchbase Bucket Management interface. On the left, there's a sidebar with navigation links: Dashboard, Servers, Buckets (which is selected and highlighted in blue), Indexes, Search, Query, and XDCR. The main area is titled "Buckets" and contains a table with the following data:

name ▾	items	resident	ops/sec	RAM used/quota	disk used		
beer-sample	7,303	100%	0	14.8MB / 100MB	30.7MB	<a href="#">Documents</a>	<a href="#">Statistics</a>
default	100,003	100%	0	22.8MB / 100MB	70.8MB	<a href="#">Documents</a>	<a href="#">Statistics</a>
travel-sample	31,592	100%	0	52.9MB / 100MB	103MB	<a href="#">Documents</a>	<a href="#">Statistics</a>

A yellow box highlights the "ADD BUCKET" button in the top right corner.

So what is the difference? The difference is that in case of henderson; there is no option to create New data Bucket/ Or Document link is enable for the beer-sample bucket only; whereas it is not the case for tiraj.

Why? [Hints --> Difference of roles]

You can try adding a user by your name as a couchbase internal user and have access to the beer-sample only as read only and log on with those credentials to verify its working.

-----End of Lab-----

## 20. Auditing for Administrators -20 Minutes(D)

In this lab, we will enable audit using Web UI. Only the full Couchbase administrators can configure auditing using the Couchbase Web Console.

We will enable Audit and will be logged in the /var/lib/couchbase/logs log folder. The log will be circular every 1 Hour.

**Following steps will be performed for enabling auditing.**

### Enable auditing

Use the check box to enable or disable auditing.

### Specify the target log directory

Specify the target directory path for storing the audit records.

### Specify log rotation

This is a log rotation time interval (in Days, Hours, or Minutes), after which the log gets rotated to the next file.

To configure auditing select: **Security > Audit:**

Enable : Audit events & write them to a log

### Audit Configuration

Auditing keeps track of important admin events occurring in Couchbase. Tracking and persisting these events is essential for any secured environment and provides evidence for suspicious/malicious activity in Couchbase.

Enable Auditing

#### Target Log Directory

/Server/var/lib/couchbase/logs

#### Log Rotation Time Interval

1 Hours

Save

Click Save.

Now let us create a bucket and delete it to confirm whether entries are there in the Audit log file.

Bucket name: test

RAM Quota : 100 MB

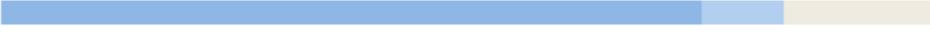
Click on Add Bucket leaving all options in default.

## Add Data Bucket

X

Name

Memory Quota in megabytes per server node  
 MB

  
other buckets (812 MB)    this bucket (100 MB)    remaining (163 MB)

Bucket Type  
 Couchbase     Memcached     Ephemeral

► Advanced bucket settings

[Cancel](#) [Add Bucket](#)

Now, You can verify the audit.log

```
#cd /opt/couchbase/var/lib/couchbase/logs  
# tail -f audit.log
```

```
{"timestamp":"2018-02-19T15:37:18.455721+05:30","real_userid":{"source":"internal","user":"couchbase"},"auditd_enabled":true,"descriptors_path":"C:\\Program Files\\Couchbase\\Server\\etc\\security","hostname":"LP000007024323","log_path":"C:\\Program Files\\Couchbase\\Server\\var\\lib\\couchbase\\logs","rotate_interval":3600,"version":1,"id":4096,"name":"configured audit daemon","description":"loaded configuration file for audit daemon"}  
{"props":{"storage_mode":"couchstore","conflict_resolution_type":"seqno","eviction_policy":"value_only","num_threads":3,"flush_enabled":false,"purge_interval":"undefined","ram_quota":104857600,"replica_index":false,"num_replicas":1},"type":"membase","bucket_name":"test","real_userid":{"source":"ns_server","user":"Administrator"}, "sessionid":"6653a8d0a1bb77ed2b58dfced9b9e33a", "remote":{"ip":"127.0.0.1","port":61108}, "timestamp":"2018-02-19T15:39:07.905+05:30", "id":8201, "name":"create bucket", "description":"Bucket was created"}  
|
```

You can find an entry that – Bucket was created.

Now, you can drop the bucket and verify its entry.

```
{"bucket_name":"test","real_userid":{"source":"ns_server","user":"Administrator"}, "sessionid":"6653a8d0a1bb77ed2b58dfced9b9e33a", "remote":{"ip":"127.0.0.1","port":61197}, "timestamp":"2018-02-19T15:41:29.108+05:30", "id":8203, "name":"delete bucket", "description":"Bucket was deleted"}  
|
```

It specified the user who has deleted the Bucket as shown above.

-----End of Lab-----

## 21. Backup and Restore – 45 Minutes (D)

This tutorial gives examples of how to use all of the commands in the 'cbbackupmgr' tool effectively.

This tutorial shows how to take backups and restore data using cbbackupmgr. This tutorial uses a cluster that contains both the travel-sample and beer-sample buckets installed and requires you to modify some of the documents in the travel-sample bucket

The only requirement for running the scripts is that you have curl installed.

### Configuring a Backup

Before getting started with cbbackupmgr you must first decide the directory where to store all of your backups. This directory is referred to as the backup archive.

A backup repository is created by using the *config* sub-command. In this tutorial we will use a backup archive located at /data/backup. The backup archive is automatically created if the directory specified is empty. Below is an example of how to create a backup repository called "cluster" which backs up all data and index definitions from all buckets in the target cluster.

Change the directory to the following folder in any of the node.

```
#cd /opt/couchbase/bin  
# ./cbbbackupmgr config --archive /data/backup --repo cluster
```

```
[root@tos bin]# pwd  
/opt/couchbase/bin  
[root@tos bin]# ./cbbbackupmgr config --archive /data/backup --repo cluster  
Backup repository `cluster` created successfully in archive `/data/backup`  
[root@tos bin]# █
```

## Backing up a Cluster

Now that you have created some backup repositories let's take a look at the backup archive to see what it looks like.

```
./cbbbackupmgr list --archive /data/backup
```

or

```
./cbbbackupmgr info --archive /data/backup
```

```
[root@tos bin]# ./cbbbackupmgr list --archive /data/backup  
Size      Items          Name  
0B        -              /  
0B        -              + cluster  
0B        -              + _
```

The list/info sub-command returns a directory print out of all of the backup repositories and backups in your backup archive. Since there are no backups yet you can just see your archives list in the output of this command. There is also information about how much disk space each folder and file contains and, if applicable, how many items are backed up in those folders/files.

Below is an example of how to take a backup on the "cluster" backup repository. Let's assume that your cluster is running on localhost.

```
# ./cbbackupmgr backup --archive /data/backup --repo cluster --host couchbase://127.0.0.1 --username Administrator --password admin123
```

```
[root@tos bin]# ./cbbackupmgr backup --archive /data/backup --repo cluster --host couchbase://127.0.0.1 --username Administrator --password admin123

Backing up to 2016-08-21T20_20_51.098674815+05_30
Copied all data in 8.06s (Avg. 403.32KB/Sec)          7303 items / 3.15MB
beer-sample      [=====] 100.00%
default         [=====] 100.00%

Backup successfully completed
[root@tos bin]#
```

```
or : ./cbackupmgr backup --archive /data/backup --repo cluster --c couchbase://127.0.0.1 --
username Administrator --password admin123
```

When the backup command is executed, by default it prints out a progress bar which is helpful to understand how long your backup will take to complete and the rate of data movement. While the backup is running, the progress bar gives an estimated time to completion, and when the backup completes, but this changes to the average backup rate. Information is also provided on the total data and items already backed up and the current rate of data movement. If the backup completes successfully, the tool prints the message "Backup completed successfully" as the last line.

Now that you have backups in your backup archive let's take a look at how the state of our backup archive has changed by using the *list* sub-command.

```
./cbackupmgr info --archive /data/backup
```

```
[root@tos bin]# ./cbbbackupmgr list --archive /data/backup
Size      Items      Name
106.10MB   -          /
106.10MB   -          + cluster
106.10MB   -          + 2016-08-21T20_20_51.098674815+05_30
56.00MB    -          + beer-sample
296B       0          bucket-config.json
56.00MB    7303       + data
56.00MB    7303       + shard_0.fdb
2B         0          full-text.json
129B       0          gsi.json
784B       1          views.json
50.10MB   -          + default
294B       0          bucket-config.json
50.10MB   0          + data
50.10MB   0          + shard_0.fdb
2B         0          full-text.json
129B       0          gsi.json
2B         0          views.json
[root@tos bin]#
```

Now that you have some backups defined, the output of the list sub-command is much more useful. You can see that the "cluster" backup repository contains one backup with a name corresponding to the time the backup was taken. That backup also contains two buckets and you can see various files in each of those backups with their size and item counts.

One of the most important features of cbbbackupmgr is that it is an incremental-only backup utility. This means that once you back up some data, you will never need to back it up again.

Let us create one document in the default bucket or in beer-sample as shown below:

Document ID : backup

```
{
  "backupId": 245,
  "Reason": "To confirm incremental back up"
}
```

<pre>backup 1 {   "backupId": 245,   "Reason": "To confirm incremental back up" 4 }</pre>	<div style="display: flex; justify-content: space-between; align-items: center;"> <span>Delete</span> <span>Save As...</span> <span>Save</span> </div> <pre>1 {   "id": "backup",   "rev": "2-   151431c0b48f00000000000000002000006",   "expiration": 0,   "flags": 33554438 6 }</pre>
---	--

After you modify some data, run the backup sub-command on the "cluster" backup repository again.

```
# ./cbbackupmgr backup --archive /data/backup --repo cluster --host couchbase://127.0.0.1 --
username Administrator --password admin123
```

```
[root@tos bin]# ./cbackupmgr backup --archive /data/backup --repo cluster --host couchbase://127.0.0.1 --username Administrator --password admin123

Backing up to 2016-08-21T20_27_44.357217074+05_30
Copied all data in 9.01s (Avg. 6.24KB/Sec)                                1 items / 56.17KB
beer-sample          [=====] 100.00%
default             [=====] 100.00%

Backup successfully completed
[root@tos bin]#
```

As you can observe above, only 1 item had been backed up.

In this backup notice that since you created one item, this is all that needs to be backed up during this run. Now list the backup archive using the list sub-command. You can see that the backup archive looks something like this:

```
# ./cbackupmgr list --archive /data/backup
or
# ./cbackupmgr info --archive /data/backup --all
```

```
[root@tos bin]# ./cbbbackupmgr list --archive /data/backup
Size      Items          Name
206.64MB   -              /
206.64MB   -          + cluster
106.42MB   -          + 2016-08-21T20_20_51.098674815+05_30
56.16MB    -          + beer-sample
296B       0              bucket-config.json
56.16MB    7303         + data
56.16MB    7303         shard_0.fdb
2B         0              full-text.json
129B       0              gsi.json
784B       1              views.json
50.26MB   -          + default
294B       0              bucket-config.json
50.26MB   0              + data
50.26MB   0              shard_0.fdb
2B         0              full-text.json
129B       0              gsi.json
2B         0              views.json
100.22MB   -          + 2016-08-21T20_27_44.357217074+05_30
50.11MB    -          + beer-sample
296B       0              bucket-config.json
50.11MB    0              + data
50.11MB    0              shard_0.fdb
2B         0              full-text.json
129B       0              gsi.json
784B       1              views.json
50.11MB   -          + default
294B       0              bucket-config.json
50.11MB    1              + data
50.11MB    1              shard_0.fdb
2B         0              full-text.json
129B       0              gsi.json
2B         0              views.json
[root@tos bin]#
```

```
[root@couchbase0 bin]# ./cbbbackupmgr info --archive /data/backup --all
Name      | UUID                                | Size      | # Repos |
backup    | e6d0282d-c9b1-4db6-8709-17240489fdac | 590.22KiB | 1        |

* Name      | Size      | # Backups |
* cluster   | 590.22KiB | 3          |

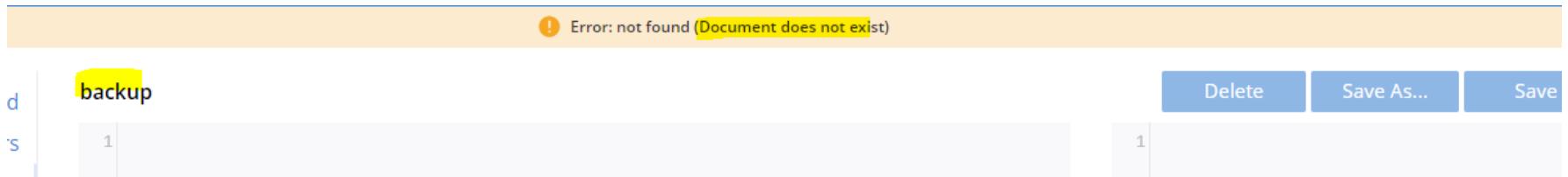
+ Backup                | Size      | Type | Source           | Cluster UUID          | Range | Events | Aliases | Comp
lete |
+ 2022-01-25T10_56_22.7185294Z | 392.79KiB | FULL | couchbase://127.0.0.1 | 9d7eeec18779bdbc7efd5ba2c6f5bfa0 | N/A   | 0      | 0       | true
|
- Bucket    | Size      | Items | Mutations | Tombstones | Views | FTS | Indexes | CBAS |
- Learning  | 392.42KiB | 2     | 2         | 0          | 0     | 0   | 0       | 0
|
- Bucket    | Size      | Items | Mutations | Tombstones | Views | FTS | Indexes | CBAS |
- beer-sample | 304B    | 0     | 0         | 0          | 0     | 0   | 0       | 0
|
+ Backup                | Size      | Type | Source           | Cluster UUID          | Range | Events | Aliases | Complete |
+ 2022-01-25T10_57_03.692474Z | 685B    | INCR | couchbase://127.0.0.1 | 9d7eeec18779bdbc7efd5ba2c6f5bfa0 | N/A   | 0      | 0       | true
|
Bucket   | Size      | Items | Mutations | Tombstones | Views | FTS | Indexes | CBAS |
```

## Restoring a Backup

Now you can delete the document from the default/beer-sample bucket before we restored from the backup. Buckets → default/beer-sample → Delete against the document row.

Click delete.

Now, you don't have that document i.e docID - backup in the default/beer-sample bucket: Using the look Up ID option.



Now let us restore back from the earlier backup.

Now that you have some backup data let's restore that data backup to the cluster. In order to restore data you just need to know the name of the backup that you want to restore. To find the name you can use the list sub-command in order to see what is in our backup archive. The backup name will always be a timestamp. For example, let's say you want to restore the 2016-08-21T20\_20\_51.098674815+05\_30 from the "cluster" backup repository. In order to do this, run the following command:

```
# ./cbbackupmgr restore --archive /data/backup --repo cluster --host http://127.0.0.1:8091 --username Administrator --password admin123 --start 2016-08-21T20_20_51.098674815+05_30 --end 2016-08-21T20_20_51.098674815+05_30 --force-updates
```

```
[root@tos bin]# ./cbbackupmgr restore --archive /data/backup --repo cluster --host http://127.0.0.1:8091 --username Administrator --password admin123 --start 2016-08-21T20_20_51.098674815+05_30 --end 2016-08-21T20_20_51.098674815+05_30 --force-updates

(1/1) Restoring backup 2016-08-21T20_20_51.098674815+05_30
Copied all data in 4.02s (Avg. 783.43KB/Sec) 7303 items / 3.06MB
beer-sample [=====] 100.00%
default [=====] 100.00%

Restore completed successfully
[root@tos bin]#
```

In the command above, notice the use of the `--start` and `--end` flags to specify the range of backups you want to restore. Since you are only restoring one backup, specify the same value for both `--start` and `--end`. The `--force-updates` flags skip Couchbase conflict resolution. This tells `cbbackupmgr` to force overwrite key-value pairs being restored even if the key-value pair on the cluster is newer than the one being restored.

Now try to look up for the doc ID : ***backup*** from the bucket  
Object won't be available since we have restored the earlier backup before that of the document deletion.



Let us restore the latest backup to retrieve the deleted document specify the second backup while restoring the backup.

```
# ./cbackupmgr restore --archive /data/backup --repo cluster --host http://127.0.0.1:8091 --username Administrator --password admin123 --start 2016-08-21T20_20_51.098674815+05_30 --end 2016-08-21T20_27_44.357217074+05_30 --force-updates
```

or

```
./cbackupmgr restore --archive /data/backup --repo cluster -c http://127.0.0.1:8091 --username Administrator --password admin123 --start 2022-01-25T10_56_22.7185294Z --end 2022-01-25T11_01_40.7923509Z --force-updates
```

```
[root@tos bin]# ./cbbackupmgr restore --archive /data/backup --repo cluster --host http://127.0.0.1:8091 --username Administrator --password admin123 --start 2016-08-21T20_20_51.098674815+05_30 --end 2016-08-21T20_27_44.357217074+05_30 --force-updates

(1/2) Restoring backup 2016-08-21T20_20_51.098674815+05_30
Copied all data in 3.03s (Avg. 1.02MB/Sec) 7303 items / 3.06MB
default [=====] 100.00%
beer-sample [=====] 100.00%

(2/2) Restoring backup 2016-08-21T20_27_44.357217074+05_30
Copied all data in 2s (Avg. 64B/Sec) 1 items / 129B
beer-sample [=====] 100.00%
default [=====] 100.00%

Restore completed successfully
[root@tos bin]#
```

Now,

You can verify the record now from the console.

default	content sample	Document ID	Look Up ID
123	{"click":"to s1","with JSON":"there are no reserved field names"}	<a href="#">Delete</a> <a href="#">Edit</a>	
backup	{"backupId":245,"Reason":"To confirm incremental back up"}	<a href="#">Delete</a> <a href="#">Edit</a>	

## Removing a Backup Repository

If you no longer need a backup repository, you can use the remove sub-command to remove the backup repository. Below is an example showing how to remove the "cluster" backup repository.

```
$ ./cbbackupmgr remove --archive /data/backup --repo cluster
```

```
[root@tos bin]# ./cbbackupmgr remove --archive /data/backup --repo cluster
Backup repository `cluster` deleted successfully from archive `/data/backup`
[root@tos bin]# ./cbbackupmgr list --archive /data/backup
  Size      Items      Name
  0B        -          /
[root@tos bin]#
```

If you now run the list sub-command you will see that the "cluster" backup repository no longer exists.

```
./cbbackupmgr list --archive /data/backup
```

or

```
./cbbackupmgr info --archive /data/backup
```

-----End of Lab-----

## 22. Full Text Search – 60 Minutes (D)

In this lab, we will explore the feature of Full Text Search in couchbase. You need to install the travel-sample bucket before proceeding ahead for this lab. ( Navigate : Settings ->Sample Buckets ) You can verify it from the web console , Bucket view.

name ▾	items	resident	ops/sec	RAM used/quota	disk used
beer-sample	7,305	100%	0	27.4MB / 100MB	13.4MB
couchmusic2	213,063	100%	0	268MB / 356MB	232MB
default	1	100%	0	22.8MB / 256MB	4.08MB
travel-sample	31,591	100%	0	65.1MB / 100MB	101MB

Let's build an index to help users find interesting results from the travel-sample bucket.

To access the **Full Text Search** screen, left-click on the **Search** tab, in the navigation bar at the left-hand side:



The **Full Text Search** screen now appears, as follows:

The screenshot shows the Couchbase Admin UI with the title 'MyCluster > Full Text Search'. On the left, there's a sidebar with navigation links: Dashboard, Servers, Buckets, **Indexes**, Analytics, Search, Query, XDCR, Security, Settings, and Logs. The 'Indexes' link is currently selected. The main area has two sections: 'Full Text Indexes' and 'Full Text Aliases'. Both sections have a table header and an 'Add' button. The 'Full Text Indexes' table has columns: index name, bucket, doc count, and indexing progress. The 'Full Text Aliases' table has columns: alias name and target indexes.

The console contains areas for the display of *indexes* and *aliases*: but both are empty, since none has yet been created.

### Create an Index

Create an Index

To create an index, left-click on the **Add Index** button, towards the right-hand side:

**Add Index**

The **Add Index** screen appears:

The screenshot shows the 'Add Index' page within the 'Full Text Search' section of the Couchbase Admin interface. On the left, a vertical sidebar lists various navigation items: Dashboard, Servers, Buckets, Indexes, Analytics, Search, Query, XDCR, Security, Settings, and Logs. The 'Indexes' item is currently selected. The main form area has the following fields:

- Name:** A text input field.
- Bucket:** A dropdown menu.
- Type Identifier:** A group of three radio button options:
  - JSON type field:
  - Doc ID up to separator:
  - Doc ID with regex:
- Type Mappings:** A section containing a list of mappings. The first item is highlighted with a grey background and checked status:  # default | dynamic.
- + Add Type Mapping:** A button to add more mappings.
- Analyzers:** A collapsed section indicated by a triangle icon.
- Custom Filters:** A collapsed section indicated by a triangle icon.
- Advanced:** A collapsed section indicated by a triangle icon.
- Index Replicas:** A dropdown menu set to 0.

At the bottom of the form are two buttons: **Create Index** (highlighted in blue) and **Cancel**.

To define a basic index on which Full Text Search can be performed, begin by entering a unique name for the index into the **Name** field, at the upper-left: for example, `travel-sample-index`. (Note that only alphanumeric characters, hyphens, and underscores are allowed for index names. Note also that the first character of the name must be an alphabetic character.) Then, use the pull-down menu provided for the **Bucket** field, at the upper-right, to specify the `travel-sample` bucket:

Name	Bucket
travel-sample-index	travel-sample

This is all you need to specify, in order to create a basic index for test and development. No further configuration is required. Note, however, that such *default indexing* is not recommended for production environments, since it creates indexes that may be unnecessarily large, and therefore insufficiently performant. After that, expand the "Type Mappings" section. There's already a "default" type mapping. If saved this index now, this index would search on every field in every document (even airline documents, and even the geolocation fields and address fields in landmark documents, which will not be helpful to a user searching for "breakfast").

So, let's narrow it down to just the content field.

Click "Add Type Mapping". Enter "landmark" as the type name. Notice that the "Type Identifier" is set to "JSON type field" with a value of "type". This is the default behavior.

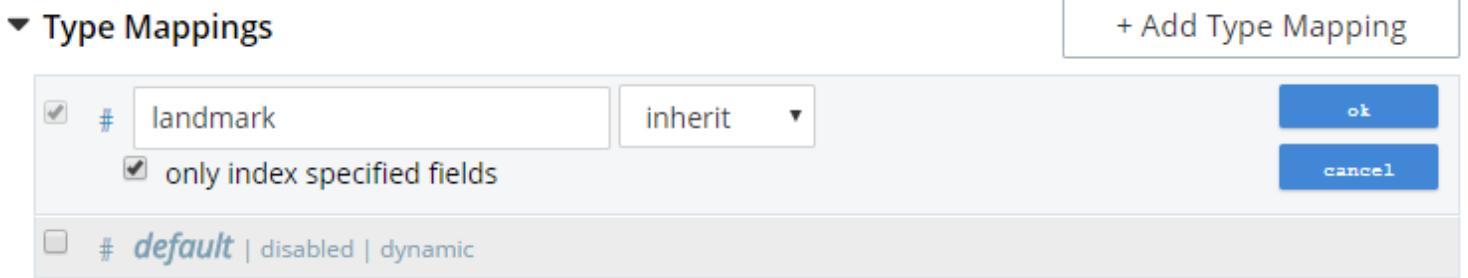
Type Identifier

JSON type field:

Doc ID up to separator:

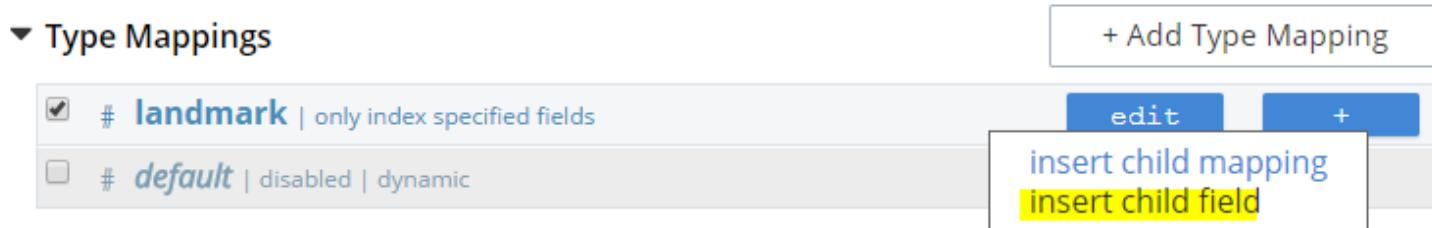
Doc ID with regex:

...



Check the box "only index specified field" and click "ok" to save. Uncheck the default.

Next, we need to add a "child field" to this index. Right now, we only want to index the content field of landmark documents. Hover over the "landmark" record's and the "+" button. Click "insert child field".



Enter "content" against the **field** and "content" against the **"searchable as"** label. Also check the "store" and all others selection as shown below. Click "ok" to save.



After that, hover over the "default" record, click "edit" and uncheck the "enabled" option. We aren't going to use the default index, so this will save time when creating the initial index.

To save your index, left-click on the **Create Index** button, near the bottom of the screen:

**Create Index**

At this point, you are returned to the **Full Text Search** screen. A row now appears, in the **Full Text Indexes** panel, for the index you have created. When left-clicked on, the row opens as follows:

travel-sample-index	travel-sample	13388	33.56%	
search this index...	Search	Delete	Clone	Edit
<a href="#">▶ Show index definition JSON</a>				

Note the percentage figure: this appears under the **indexing progress** column, and is incremented in correspondence with the build-progress of the index. When 100% is reached, the build is complete, and the index ready for use.

Once the new index has been built, it supports Full Text Searches performed by all available means: The Console UI, the Couchbase REST API, and the Couchbase SDK.

### Perform a Query

To perform a query, simply type a term into the interactive text-field that appears to the left of the **Search** button on the row for the index you have created. For example, `restaurant`. Then, left-click on the **Search** button:



A **Search Results** page now appears, featuring documents that contain the specified term:

restaurant   show advanced query settings  
[full text query syntax help](#)

**Results for travel-sample-index**

Show Scoring 911 results (28ms server-side)

1. <a href="#">landmark_33429</a>	
2. <a href="#">landmark_20417</a>	
3. <a href="#">landmark_21238</a>	
4. <a href="#">landmark_8642</a>	
5. <a href="#">landmark_25189</a>	
6. <a href="#">landmark_1159</a>	
7. <a href="#">landmark_26114</a>	
8. <a href="#">landmark_37331</a>	
9. <a href="#">landmark_17578</a>	
10. <a href="#">landmark_24618</a>	

1 2 3 4 5 »

By left-clicking on any of the displayed document IDs, you bring up a display that features the entire contents of the document.

## Advanced Settings and Other Features

On the **Search Results** page, to the immediate right of the **Search** button, at the top of the screen, appears the **show advanced query settings** checkbox. Check this to display the advanced settings:

The screenshot shows the search interface with the 'show advanced query settings' checkbox checked. Below it, three input fields are visible: 'Timeout (msecs)' with value '0', 'Consistency Level' (empty), and 'Consistency Vectors' with value '0'. A 'JSON for Query Request' section displays the following JSON query:

```
{  
  "explain": true,  
  "fields": [  
    "*"  
  ],  
  "highlight": {},  
  "query": {  
    "query": "restaurant"  
  }  
}
```

A 'Copy to Clipboard' button is located to the right of the JSON code.

### Results for travel-sample-index

Show Scoring

911 results (19ms server-side)

1. [landmark\\_33429](#)

Three interactive text-fields now appear underneath the **Search** panel: **Timeout (msecs)**, **Consistency Level**, and **Consistency Vector**. Additionally, the **JSON for Query Request** panel displays the submitted query in JSON format. Note the **show command-line curl example** checkbox, which when checked, adds to the initial JSON display, to form a completed curl command:

JSON for Query Request  show command-line curl example

```
curl -XPOST -H "Content-Type: application/json" \
http://localhost:8094/api/index/travel-sample-index/query \
-d '{
  "explain": true,
  "fields": [
    "*"
  ],
  "highlight": {},
  "query": {
    "query": "restaurant"
  },
  "ctl": {
    "consistency": {
      "level": "at_plus"
    }
  }
}'
```

This example can be copied by means of the **Copy to Clipboard** button, pasted (for example) into a standard console-window, and executed against the prompt. This feature therefore provides a useful means of extending experiments initially performed with the UI into a subsequent console-based,

script-based, or program-based context. Note also the **Show/Explain Scoring** checkbox that appears prior to the entries in the **Results for travel-sample-index** panel. When this is checked, scores for each document in the list are provided. For example:

Results for travel-sample-index

Show Scoring      911 results (9ms server-side)

1. [landmark\\_33429](#)

Scoring

- 0.812 - product of:
  - 0.812 - sum of:
    - 0.812 - product of:
      - 0.812 - sum of:
        - 0.812 - weight(\_all:restaurant^1.000000 in landmark\_33429), product of:
          - 1.000 - queryWeight(\_all:restaurant^1.000000), product of:
            - 1.000 - boost
          - 4.545 - idf(docFreq=911, maxDocs=31591)
        - 0.220 - queryNorm
      - 0.812 - fieldWeight(\_all:restaurant in landmark\_33429), product of:
        - 1.732 - tf(termFreq(\_all:restaurant)=3
        - 0.103 - fieldNorm(field=\_all, doc=landmark\_33429)
        - 4.545 - idf(docFreq=911, maxDocs=31591)
      - 1.000 - coord(1/1)
    - 1.000 - coord(1/1)

Finally, note the **full text query syntax help** link that now appears under the **Search** interactive text-field:



This link takes the user to a [page](#) of information on *Query String* Full Text Search queries. Such queries can be specified in the **Search** interactive text-field, thereby allowing searches of considerable complexity to be accomplished within the Couchbase Web Console.

----- Lab Ends Here -----

## 23. Eventing service – 60 Minutes(D)

Given a legacy document set containing attributes whose format makes them difficult to search on. In order to correct this search deficiency new searchable attributes will be added to the document. These new attributes related to and can be calculated from the original attributes. On any mutation (a document creation or modification) the new attributes should also be created (or updated). We will perform a data enrichment using Eventing services of Couchbase.

**Implementation:** Implementation: Create JavaScript function that contains an **OnUpdate** handler. The handler listens for mutations or data-changes within a specified "Listen To Location" (or source collection). When any document within the collection is created or modified, the handler executes a user-defined routine. In this example, if the created or altered document contains two specifically named fields containing IP addresses (these respectively corresponding to the beginning and end of an address-range), the handler-routine converts each of the IP addresses to an integer and upserts them as new fields in the document.

Proceed as follows: (6.0 or less than 7)

1. Create three buckets **source, target and metadata** buckets. The target bucket contains documents that will be created after the Function execution step. Don't add any documents explicitly to the target bucket.

To create a bucket, refer to [Create a Bucket](#).

	name ▾	items	resident	ops/sec	RAM used/quota	disk used	
	default	0	100%	0	20.5MB / 100MB	4.06MB	<a href="#">Documents Statistics</a>
	metadata	0	100%	0	20.5MB / 100MB	4.05MB	<a href="#">Documents Statistics</a>
	source	0	100%	0	20.5MB / 100MB	4.06MB	<a href="#">Documents Statistics</a>
	target	0	100%	0	20.5MB / 100MB	4.05MB	<a href="#">Documents Statistics</a>

1. From the Couchbase Web Console > Buckets page, click **source** bucket.
2. In the **source** bucket screen:
  1. Click Add Document.
  2. In the Add Document window, specify the name **SampleDocument** as the New Document ID
  3. Click Save.
- b. In the Edit Document dialog, paste the following within the edit panel.

{

```
"country": "AD",
"ip_end": "5.62.60.9",
"ip_start": "5.62.60.1"
}
```

- c. Click Save. This step concludes all required preparations.

The screenshot shows the Tos.Couchbase Documents page. The top navigation bar includes a logo, the bucket name 'Tos.Couchbase > Documents', and links for 'CLASSIC EDITOR' and 'ADD DOCUMENT'. On the left, a sidebar lists navigation items: Dashboard, Servers, Buckets, XDCR, Security, Settings, and Logs. The main content area has a 'Bucket' dropdown set to 'source', 'Limit' at 10, 'Offset' at 0, and a 'Where' input field. A 'Retrieve Docs' button is present. Below this, a message says '1 Results for source, limit: 10, offset: 0'. The results table shows one row for 'SampleDocument' with ID 'id'. The document content is shown as a JSON object: {"country": "AD", "ip\_end": "5.62.60.9", "ip\_start": "5.62.60.1"}. Action buttons for edit, delete, and copy are visible next to the document row. Navigation buttons for 'Prev Batch' and 'Next Batch' are also present.

## Procedure

Proceed as follows:

1. From the Couchbase Web Console > ***Eventing*** page, click ADD FUNCTION, to add a new Function.
2. In the ADD FUNCTION dialog, for individual Function elements provide the below information:
  - For the ***Listen To Location (Source)*** Bucket drop-down, select the ***source*** option that was created for this purpose.
  - For the ***Eventing Storage (Metadata)*** Bucket drop-down, select the ***metadata*** option that was created for this purpose.
  - Enter ***enrich\_ip\_nums*** as the name of the Function you are creating in the Function\*Name\* text-box.
  - Enter Enrich a document, converts IP Strings to Integers that are stored in new attributes, in the Description text-box.
  - For the Settings option, use the default values.
  - For the Bindings option, specify target as shown below.

**Description** (optional)  
Enter Enrich a document, converts IP Strings to Integers that are stored in new attributes,

▶ Settings

**Bindings**

+ -

✓ bucket alias ▾ src

**Bucket** bucket.scope.collection      **Access**

source ▾ . \_default ▾ . \_default ▾ . read only ▾

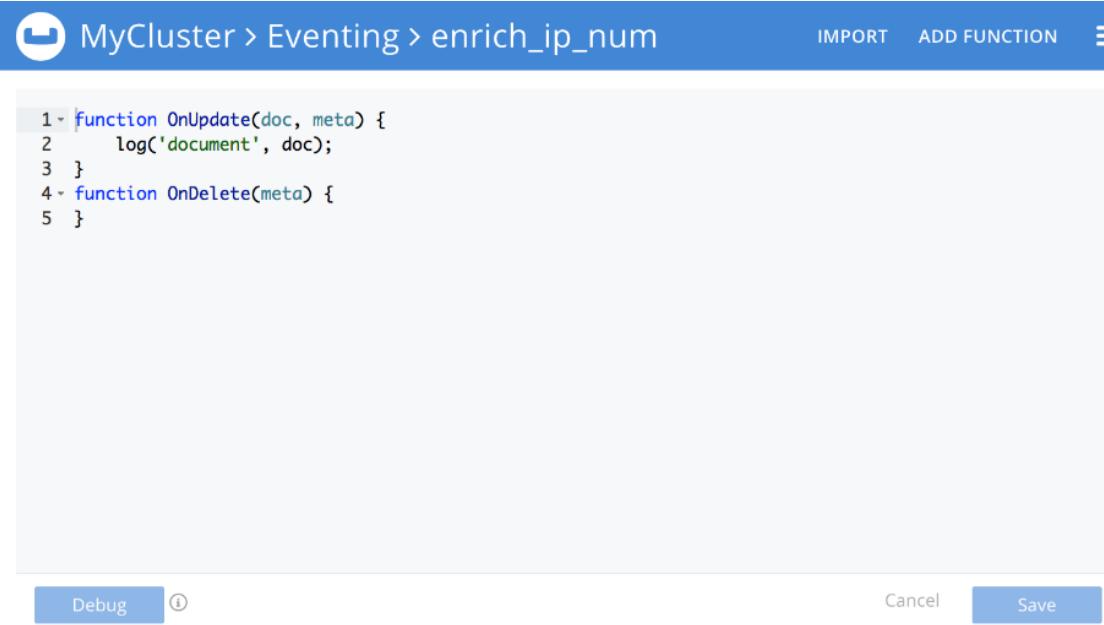
✓ bucket alias ▾ target

**Bucket** bucket.scope.collection      **Access**

target ▾ . \_default ▾ . \_default ▾ . read and write ▾

[Cancel](#) [Next: Add Code](#)

3. After providing all the required information in the ADD FUNCTION dialog, click Next: Add Code. The **enrich\_ip\_nums** dialog appears. The **enrich\_ip\_nums** dialog initially contains a placeholder code block. You will substitute your actual **enrich\_ip\_nums** code in this block.



The screenshot shows the Couchbase Eventing interface. The top navigation bar includes a cluster icon, the name 'MyCluster', and sections for 'Eventing' and 'enrich\_ip\_num'. On the right, there are buttons for 'IMPORT' and 'ADD FUNCTION'. Below the navigation is a code editor window. The code area contains the following placeholder JavaScript:

```
1 - function OnUpdate(doc, meta) {  
2     log('document', doc);  
3 }  
4 - function OnDelete(meta) {  
5 }
```

At the bottom of the code editor, there are three buttons: 'Debug', 'Cancel', and 'Save'.

4. Copy the following Function, and paste it in the placeholder code block of the **enrich\_ip\_nums** dialog:

```
function OnUpdate(doc, meta) {
```

```
log('document', doc);
doc["ip_num_start"] = get_numip_first_3_octets(doc["ip_start"]);
doc["ip_num_end"]   = get_numip_first_3_octets(doc["ip_end"]);
target[meta.id]=doc;
}

function get_numip_first_3_octets(ip)
{
var return_val = 0;
if (ip)
{
var parts = ip.split('.');
//IP Number = A x (256*256*256) + B x (256*256) + C x 256 + D
return_val = (parts[0] * (256*256*256)) + (parts[1] * (256*256)) + (parts[2] * 256) +
parseInt(parts[3]);
return return_val;
}
}
```

After pasting, the screen appears as displayed below:

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

```

1+ function OnUpdate(doc, meta) {
2    log('document', doc);
3    doc["ip_num_start"] = get_numip_first_3_octets(doc["ip_start"]);
4    doc["ip_num_end"]   = get_numip_first_3_octets(doc["ip_end"]);
5    tgt[meta.id]=doc;
6 }
7 function get_numip_first_3_octets(ip)
8 {
9     var return_val = 0;
10    if (ip)
11    {
12        var parts = ip.split('.');
13
14        //IP Number = A x (256*256*256) + B x (256*256) + C x 256 + D
15        return_val = (parts[0]*(256*256*256)) + (parts[1]*(256*256)) + (parts[2]*256) + parseInt(parts[3]);
16        return return_val;
17    }
18 }
19
20+ function OnDelete(meta) {
21 }
```

The ***OnUpdate*** routine specifies that when a change occurs to data within the bucket, the routine ***get\_numip\_first\_3\_octets*** is run on each document that contains ***ip\_start*** and ***ip\_end***. A new document is created whose data and metadata are based on those of the document on which *get\_numip\_first\_3\_octets* is run; but with the addition of ***ip\_num\_start*** and ***ip\_num\_end*** data-fields, which contain the numeric values returned by *get\_numip\_first\_3\_octets*. The *get\_numip\_first\_3\_octets* routine splits the IP

address, converts each fragment to a numeral, and adds the numerals together, to form a single value; which it returns.

5. Click Save.
6. To return to the Eventing screen, click Eventing and click on the newly created Function name. The Function ***enrich\_ip\_nums*** is listed as a defined Function.

function name	status	Actions
enrich_ip_num	⚠️ undeployed ⚠️ paused	Settings
enrich_ip_nums		Delete Export Deploy Edit JavaScript

7. Click Deploy.
8. From the Confirm Deploy Function dialog, click Deploy Function. From this point, the defined Function is executed on all existing documents and on subsequent mutations. After a few moments the status of the function should be deployed as shown below

function name	status	Actions
enrich_ip_nums	deployed	Settings
Enter Enrich a document, converts IP Strings to Integers that are stored in new attributes		
Delete Export Undeploy Edit JavaScript		

9. To check results of the deployed Function, click the Documents tab.

10. Select **target** bucket from the Bucket drop-down. As this shows, a version of **SampleDocument** has been added to the target bucket. It contains all the attributes of the original document, with the addition of **ip\_num\_start** and **ip\_num\_end**; which contain the numeric values that correspond to **ip\_start** and **ip\_end**, respectively. Additional documents added to the **source** bucket, which share the **ip\_start** and **ip\_end** attributes, will be similarly handled by the defined Function: creating such a document, and changing any attribute in such a document both cause the Function's execution.

The screenshot shows the Couchbase Admin UI interface for managing buckets. At the top, there are input fields for 'Bucket' (set to 'target'), 'Limit' (10), 'Offset' (0), 'Document ID' (empty), and 'Where' (e.g., 'meta().id = "some\_id" ar'). A 'Retrieve Docs' button is also present. Below the search bar, it says '1 Results for target, limit: 10, offset: 0'. There are 'simple' and 'spreadsheet' view options, with 'simple' selected. Navigation buttons for '< Prev Batch' and 'Next Batch >' are shown. The results table has columns for '\_id' and '\_source'. One result is listed: 'SampleDocument' with the value '{"country":"AD","ip\_end":"5.62.60.9","ip\_start":"5.62.60.1","ip\_num\_start":87964673,"ip\_num\_end":'. The 'ip\_num\_start' and 'ip\_num\_end' values are underlined in red.

Create two documents as shown below in the Source bucket.

The screenshot shows the Couchbase Admin UI interface for document retrieval. At the top, there are filters for Bucket (source), Limit (10), Offset (0), Document ID (empty), and Where (e.g., 'meta().id = "some\_id"'). Below these are buttons for 'Retrieve Docs' and 'Where' (simple/spreadsheet). The results section displays 3 Results for source, limit: 10, offset: 0. It includes a toggle between simple and spreadsheet views, and buttons for < Prev Batch and Next Batch >. The results table has columns for id, name, and value. The first row is SampleDocument with value {"country":"AD","ip\_end":"5.62.60.9","ip\_start":"5.62.60.1"}. The second row is SampleDocument\_1 with value {"country":"AD","ip\_end":"5.62.60.9","ip\_start":"5.62.60.1"}. The third row is SampleDocument\_2 with value {"Name":"henry","Learning":"Eventing"}. The rows for SampleDocument\_1 and SampleDocument\_2 have red horizontal lines underneath them.

id		
SampleDocument	<a href="#">Edit</a> <a href="#">View</a> <a href="#">Delete</a> <a href="#">File</a>	{"country":"AD","ip_end":"5.62.60.9","ip_start":"5.62.60.1"}
SampleDocument_1	<a href="#">Edit</a> <a href="#">View</a> <a href="#">Delete</a> <a href="#">File</a>	{"country":"AD","ip_end":"5.62.60.9","ip_start":"5.62.60.1"}
SampleDocument_2	<a href="#">Edit</a> <a href="#">View</a> <a href="#">Delete</a> <a href="#">File</a>	{"Name":"henry","Learning":"Eventing"}

## Verify in Document tab

This screenshot shows the same document retrieval interface as above, but with a different bucket selected: target. The results show 3 Results for target, limit: 10, offset: 0. The data structure is identical to the previous screenshot, with SampleDocument, SampleDocument\_1, and SampleDocument\_2 entries. The rows for SampleDocument\_1 and SampleDocument\_2 have red horizontal lines underneath them.

id		
SampleDocument	<a href="#">Edit</a> <a href="#">View</a> <a href="#">Delete</a> <a href="#">File</a>	{"country":"AD","ip_end":"5.62.60.9","ip_start":"5.62.60.1","ip_num_start":
SampleDocument_1	<a href="#">Edit</a> <a href="#">View</a> <a href="#">Delete</a> <a href="#">File</a>	{"country":"AD","ip_end":"5.62.60.9","ip_start":"5.62.60.1","ip_num_start":
SampleDocument_2	<a href="#">Edit</a> <a href="#">View</a> <a href="#">Delete</a> <a href="#">File</a>	{"Name":"henry","Learning":"Eventing"}

As you can see, the SampleDocument\_2 document doesn't have the additional attributes of eventing.

### Couchbase 7.0

For this example, two (2) buckets '**bulk**' and '**rr100**' are required where the later is intended to be 100% resident. Create the buckets with a minimum size of 100MB. Within the buckets we need three (3)

[bucket.scope.collection]keyspaces '**bulk.data.source**', '**bulk.data.target**', and '**rr100.eventing.metadata**' (we loosely follow this [organization](#)).

The Eventing Storage keyspace, in this case '**rr100.eventing.metadata**', is for the sole use of the Eventing system, do not add, modify, or delete documents from it. In addition do not drop or flush or delete the containing bucket (or delete this collection) while you have any deployed Eventing functions. In a single tenancy deployment this collection can be shared with other Eventing functions.

- 18 Access the **Couchbase Web Console > Buckets** page and click the **Scopes and Collections** link of the **bulk** bucket.

Click **Documents** in the upper right banner for the **data** scope.

Select the keyspace **bulk**, **data**, **source**

You should see no user records.

Click **Add Document** in the upper right banner

For the **ID** in the **Create New Document** dialog specify **SampleDocument** ID [  
    SampleDocument       ]

For the document body in the **Create New Document** dialog, the following text is displayed:{

```
"click": "to edit",
"with JSON": "there are no reserved field names"
}
```

replace the above text with the following JSON document via a cut-n-paste

```
{
"country": "AD",
```

```
"ip_start": "5.62.60.1",
"ip_end": "5.62.60.9"
}
```

Click **Save**.

From the **Couchbase Web Console > Eventing** page, click **ADD FUNCTION**, to add a new Function. The **ADD FUNCTION** dialog appears.

- 19 In the **ADD FUNCTION** dialog, for individual Function elements provide the below information:

For the **Listen To Location** drop-down, select **bulk, data, source** as the keyspace.

For the **Eventing Storage** drop-down, select **rr100, eventing, metadata** as the keyspace.

Enter **case\_1\_enrich\_ips** as the name of the Function you are creating in the **Function Name** text-box.

Leave the "Deployment Feed Boundary" as Everything.

[Optional Step] Enter text **On mutation create a new document in a different collection with additional fields**, in the **Description** text-box.

For the **Settings** option, use the default values.

For the **Bindings** option, add two bindings.

For the first binding, select "bucket alias", specify **src** as the "alias name" of the collection, select **bulk, data, source** as the associated keyspace, and select "read only" for the access mode.

For the second binding, select "bucket alias", specify **tgt** as the "alias name" of

the collection, select **bulk**, **data**, and **target** as the associated keyspace, and select "read and write" for the access mode.

After configuring your settings the **ADD FUNCTION** dialog should look like this:

### Add Function

**Listen To Location** bucket.scope.collection ⓘ  
bulk ▾ data ▾ source ▾

**Eventing Storage** bucket.scope.collection ⓘ  
rr100 ▾ eventing ▾ metadata ▾

System data stored in this location will have the document ID prefixed with **eventing**

**Function Name**  
case\_1\_enrich\_ips

**Deployment Feed Boundary** ⓘ  
Everything ▾

**Description** (optional)  
On mutation create a new document in a different collection with additional fields

▶ Settings

**Bindings** + -

**src** Bucket bucket.scope.collection Access  
bulk ▾ . data ▾ . source ▾ read only ▾

**tgt** Bucket bucket.scope.collection Access  
bulk ▾ . data ▾ . target ▾ read and write ▾

**Cancel** **Next: Add Code**

- 20 After providing all the required information in the **ADD FUNCTION** dialog, click **Next: Add Code**. The **case\_1\_enrich\_ips** dialog appears.

The **case\_1\_enrich\_ips** dialog initially contains a placeholder code block. You will substitute your actual **case\_1\_enrich\_ips code** in this block.



The screenshot shows the Couchbase Eventing Function Editor interface. The left sidebar has links for Dashboard, Servers, Buckets, Backup, XDCR, Security, and Settings. The main area is titled "Function Editor" and contains the following placeholder code:

```
1 function OnUpdate(doc, meta) {  
2     log("Doc created/updated", meta.id);  
3 }  
4  
5 function OnDelete(meta, options) {  
6     log("Doc deleted/expired", meta.id);  
7 }
```

At the top right, there are buttons for SETTINGS, IMPORT, and ADD FUNCTION. Below the code editor, there is a link "< back to Eventing" and a magnifying glass icon for search.

Copy the following Function, and paste it in the placeholder code block of  
**case\_1\_enrich\_ips** dialog.JAVASCRIPTCopy

```
function OnUpdate(doc, meta) {
    log('document', doc);
    doc["ip_num_start"] = get_numip_first_3_octets(doc["ip_start"]);
    doc["ip_num_end"] = get_numip_first_3_octets(doc["ip_end"]);
    tgt[meta.id]=doc;
}
function get_numip_first_3_octets(ip) {
    var return_val = 0;
    if (ip) {
        var parts = ip.split('.');
        //IP Number = A x (256*256*256) + B x (256*256) + C x 256 + D
        return_val = (parts[0]*(256*256*256)) + (parts[1]*(256*256)) + (parts[2]*256) +
            parseInt(parts[3]);
        return return_val;
    }
}
```

After pasting, the screen appears as displayed below:

The screenshot shows the Couchbase Eventing Function Editor interface. The left sidebar contains navigation links: Dashboard, Servers, Buckets, Backup, XDCR, Security, Settings, Logs, Documents, Query, Indexes, Search, and a 'More' section. The main area is titled 'Function Editor' and displays two functions:

```

1 function OnUpdate(doc, meta) {
2   log('document', doc);
3   doc["ip_num_start"] = get_numip_first_3_octets(doc["ip_start"]);
4   doc["ip_num_end"] = get_numip_first_3_octets(doc["ip_end"]);
5   tgt[meta.id]=doc;
6 }
7 function get_numip_first_3_octets(ip) {
8   var return_val = 0;
9   if (ip) {
10     var parts = ip.split('.');
11     //IP Number = A x (256*256*256) + B x (256*256) + C x 256 + D
12     return_val = (parts[0]*(256*256*256)) + (parts[1]*(256*256)) + (parts[2]*256) + parseInt(parts[3]);
13     return return_val;
14   }
15 }

```

At the top right, there are 'SETTINGS', 'IMPORT', and 'ADD FUNCTION' buttons, and a search icon.

**Click Save and Return.**

- 21 The **OnUpdate** routine specifies that when a change occurs to data within the bucket, the routine **get\_numip\_first\_3\_octets** is run on each document that contains **ip\_start** and **ip\_end**. A new document is created whose data and metadata are based on those of the document on which **get\_numip\_first\_3\_octets** is run; but with the addition of **ip\_num\_start** and **ip\_num\_end** data-fields, which contain the numeric values returned by **get\_numip\_first\_3\_octets**. The **get\_numip\_first\_3\_octets** routine splits the IP address, converts each fragment to a numeral, and adds the numerals together, to form a single value; which it returns.

- 22 From the **Eventing** screen, click the **case\_1\_enrich\_ips** function to select it, then click **Deploy**.

The screenshot shows the Couchbase Eventing interface. On the left, there's a sidebar with links: Buckets, Backup, XDCR, Security, and Settings. The main area displays a table for the 'case\_1\_enrich\_ips' function. The table has the following columns: Name (case\_1\_enrich\_ips), Source (bulk.data.source), Status (undeployed), and Actions (Settings, Delete, Export, Deploy, Pause, Edit JavaScript). The 'Edit JavaScript' button is highlighted in blue. Below the table, there's a note: 'On mutation create a new document in a different collection with additional fields'.

Buckets	case_1_enrich_ips	bulk.data.source	undeployed	Settings
Backup	On mutation create a new document in a different collection with additional fields			
XDCR				
Security				
Settings				

**case\_1\_enrich\_ips**

bulk.data.source

undeployed

Settings

Delete Export Deploy Pause Edit JavaScript

On mutation create a new document in a different collection with additional fields

In the **Confirm Deploy Function** Click **Deploy Function**.

- 23 The Eventing function is deployed and starts running within a few seconds. From this point, the defined Function is executed on all existing documents and will also more importantly it will also run on subsequent mutations.

- 24 To check the results of the deployed Eventing Function:  
Access the **Couchbase Web Console > Buckets** page and click the **Scopes and Collections** link of the **bulk** bucket.  
Click **Documents** in the upper right banner for the **data** scope.  
Select the keyspace **bulk**, **data**, **target**  
Edit the document and you will see a duplicate of the source bucket but without two new calculated fields as follows:

```
{  
  "country": "AD",  
  "ip_end": "5.62.60.9",  
  "ip_start": "5.62.60.1",  
  "ip_num_start": 87964673,  
  "ip_num_end": 87964681  
}
```

Click **Cancel** to close the editor.

- 25 Because our Eventing Function is deployed it will continue to process all new mutations, let's test this out.

Access the **Couchbase Web Console > Buckets** page and click the **Scopes and Collections** link of the **bulk** bucket.

Click **Documents** in the upper right banner for the **data** scope.

Select the keyspace **bulk**, **data**, **source**

You should see one user record (the one we entered at the beginning of this procedure).

Click **Add Document** in the upper right banner

For the **ID** in the **Create New Document** dialog specify **AnotherSampleDocument**  
ID [ AnotherSampleDocument ]

For the document body in the **Create New Document** dialog, the following text is displayed:

```
{  
  "click": "to edit",  
  "with JSON": "there are no reserved field names"  
}
```

replace the above text with the following JSON document via a cut-n-paste

```
{  
  "country": "RU",  
  "ip_start": "7.12.60.1",  
  "ip_end": "7.62.60.9"  
}
```

**Click Save.**

The screenshot shows the Couchbase Web Console interface. At the top, there are dropdown menus for Keyspace (set to 'bulk'), Bucket (set to 'data'), Scope (set to 'source'), and Collection (set to 'bulk'). Below these are input fields for Limit (10), Offset (0), Document ID (optional...), and N1QL WHERE (no indexes available...). A 'Retrieve Docs' button is also present. Underneath the search bar, it says '2 Results for bulk.data.source, limit: 10, offset: 0'. There is a toggle switch for 'enable field editing'. On the right, there are links for '< prev batch' and 'next batch >'. The main area displays two documents:

id	Document	Content
	AnotherSampleDocument	{"country": "RU", "ip_start": "7.12.60.1", "ip_end": "7.62.60.9"}
	SampleDocument	{"country": "AD", "ip_start": "5.62.60.1", "ip_end": "5.62.60.9"}

- 26 To check results (**which were updated in real time**) by the deployed Eventing Function:

Access the **Couchbase Web Console > Buckets** page and click the **Scopes and Collections** link of the **bulk** bucket.  
Click **Documents** in the upper right banner for the **data** scope.

Select the keyspace **bulk**, **data**, **target**

Edit the newly created document and you will see a duplicate of the source bucket but without two new calculated fields as follows:

```
{
  "country": "RU",
  "ip_end": "7.62.60.9",
  "ip_start": "7.12.60.1",
  "ip_num_start": 118242305,
  "ip_num_end": 121519113
}
```

Click **Cancel** to close the editor.

----- Lab Ends -----

## 24. Analytics service(s) - 60 Minutes(D)

Go to Analytics Query Editor.

A newly created Analytics instance starts out *empty*. That is, it contains no data other than the Analytics system catalogs. These system catalogs live in a special dataverse called the Metadata dataverse. If you want to know what dataverses have been defined so far, you can query the Dataverse dataset in the Metadata dataverse as shown below:

As shown below: There is only One Dataverse.

```
SELECT * FROM Metadata.`Dataverse`;
```

The output will look as follows on a freshly installed system:

```
[  
 {  
   "Dataverse": {  
     "DataverseName": "Default",  
     "DataFormat": "org.apache.asterix.runtimeformats.NonTaggedDataFormat",  
     "Timestamp": "Fri Aug 17 09:47:38 BST 2018",  
     "PendingOp": 0  
   }  
 },  
 {  
   "Dataverse": {  
     "DataverseName": "Metadata",  
     "DataFormat": "org.apache.asterix.runtimeformats.NonTaggedDataFormat",  
     "Timestamp": "Fri Aug 17 09:47:38 BST 2018",  
     "PendingOp": 0  
   }  
 }  
 ]
```

The output above shows that a fresh Analytics instance starts out with two dataverses, one called *Metadata* (the system catalog) and one called *Default*(available for holding data). This sample scenario deals with the information about beers and breweries, which you'll need to get from Couchbase Server. To keep things simple, we will use the *Default* dataverse here. That means your first task is to tell Analytics about the Couchbase Server data that you want to shadow and the datasets where you want it to live.

If you want to see what Analytics scopes have been defined so far for your user data, the simplest way is to look at the *Analytics Scopes, Links, & Collections* panel on the right-hand side of the Analytics Workbench in the Couchbase Server Web Console. Initially you will see one Analytics scope, named *Default*, which has no collections yet but is available for holding user collections when no other scope has been specified. That panel is organized by scope; within each scope you will see one or more links — references to Couchbase Server clusters — and under each link is a list of the Analytics collections in this Analytics service instance that are coming from Data service collections in the referenced cluster. Initially there is just a Local link, referring to the cluster where this Analytics service instance is running.

Our sample scenario here deals with travel information. You should start by installing the travel-sample bucket (using the Couchbase Web Console **Settings Sample Buckets** tab) in order to try all of the steps for yourself. Your first task will be to tell Analytics about the Data service collections that you want it to shadow and the Analytics collections where you want the data to live. There is quite a bit of mapping flexibility available there, but we will keep things simple for now. The following series of Analytics DDL statements shows you how to tell Analytics to track the collections of interest in the travel-sample bucket in the Data service, which is where the desired travel data resides. These statements instruct Analytics to shadow the Data service collections in a set of corresponding Analytics collections:

```
ALTER COLLECTION `travel-sample`.inventory.airport ENABLE  
ANALYTICS;
```

```
ALTER COLLECTION `travel-sample`.inventory.airline ENABLE ANALYTICS;
ALTER COLLECTION `travel-sample`.inventory.route ENABLE ANALYTICS;
ALTER COLLECTION `travel-sample`.inventory.landmark ENABLE ANALYTICS;
ALTER COLLECTION `travel-sample`.inventory.hotel ENABLE ANALYTICS;
```

In slightly more detail, these ALTER COLLECTION statements operate on the airport, airline, route, landmark, and hotel collections, respectively, each of which resides in the inventory scope of the travel-sample bucket in the local cluster's Data service. Since your Analytics instance started out empty, the first of these statements will have an extra side-effect — it will also create a corresponding scope in Analytics where your Analytics collections will reside. The statements then create the target Analytics collections in Analytics for the data of interest. (If you prefer, the outcome same can be achieved graphically by clicking on the Map From Data Service Collections button near the top right in the UI.) The resulting Analytics collections are separate copies of your data that will be hash-partitioned (sharded) across all of the nodes running instances of the Analytics service. The hash partitioning sets the stage for the parallel processing that Analytics employs when processing complex analytical queries.

The shadowing relationship of these Analytics collections to the data in Couchbase Server begins immediately upon execution of each ALTER COLLECTION statement (as you may have already inferred from the activity on the right-hand side of the Analytics Workbench). As each statement is run, Analytics begins ingesting its own copy of the corresponding Couchbase Server collection and continuously monitors it for changes. To do so, it makes

use of a link (in this case the Local link) between your Analytics scope and the Data service of the Couchbase Server cluster where the to-be-shadowed data resides. In this case, the data of interest is in the local server (i.e., it is managed by the Data service in the same cluster). The importance of the fact that Analytics has its own copy of the data cannot be over-emphasized — this is what provides the *performance isolation* that makes it safe to run potentially expensive queries at the same time the Data Service is servicing your operational applications. This is also why it is crucial for the Analytics Service to run on its own nodes within your Couchbase Server cluster; no other services should ever be co-located with Analytics in production.

You can see that your Analytics collections are all there and being populated by looking at the **Analytics Scopes,Links, & Collections** panel. At this point, assuming that you are just getting started, you will see only your five new Analytics collections listed there. You can now have a look at your Analytics data.

The following query asks Analytics for the current number of airlines:

Analytics Scopes, Links, & Collections

Map From Data Service

**Default** + remote link

**Local cb local %** + collection

▶ breweries

**travel-sample.inventory** + remote link

**Local cb local %** + collection

▶ airline  
progress 0%

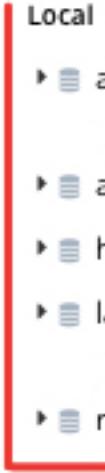
▶ airport

▶ hotel

▶ landmark  
progress 0%

▶ route

[Hide empty analytics scopes](#)



```
SELECT VALUE COUNT(*) FROM `travel-sample`.inventory.airline;
```

It returns:

JSONCopy

```
[  
  187  
]
```

Since always specifying the scope explicitly for a collection can be tedious, you can use the **query context** drop-down list at the top right of the Query Editor to select `travel-sample.inventory` as the default scope for the collection names in your queries:

✓ query context

Default

travel-sample.inventory

You can then simply enter the following to obtain the same result as above:

N1QLCopy

```
SELECT VALUE COUNT(*) FROM airline;
```

The next query retrieves a sample airline:

N1QLCopy

```
SELECT VALUE al FROM airline al ORDER BY name LIMIT 1;
```

It returns:

JSONCopy

```
[
```

```
{  
  "id": 10,  
  "type": "airline",  
  "name": "40-Mile Air",  
  "iata": "Q5",  
  "icao": "MLA",  
  "callsign": "MILE-AIR",  
  "country": "United States"  
}  
]
```

The following query asks Analytics for the number of airports:

N1QLCopy

```
SELECT VALUE COUNT(*) FROM airport;
```

It returns:

JSONCopy

```
[  
  1968  
]
```

The following query retrieves a sample airport:

N1QLCopy

```
SELECT VALUE ap FROM airport ap ORDER BY id LIMIT 1;
```

It returns:

JSONCopy

```
[
```

```
{
  "id": 465,
  "type": "airport",
  "airportname": "Belfast Intl",
  "city": "Belfast",
  "country": "United Kingdom",
  "faa": "BFS",
  "icao": "EGAA",
  "tz": "Europe/London",
  "geo": {
    "lat": 54.6575,
    "lon": -6.215833,
    "alt": 268
  }
}
]
```

You can sample the other three travel-sample collections as well to get a feel for the data in each of them before writing more interesting queries.

Route:

N1QLCopy

```
SELECT VALUE rt FROM route rt ORDER BY id LIMIT 1;
```

JSONCopy

```
[{
}
```

```
"id": 117,
"type": "route",
"airline": "2L",
"airlineid": "airline_2750",
"sourceairport": "BOD",
"destinationairport": "ZRH",
"stops": 0,
"equipment": "100",
"schedule": [
  {
    "day": 0,
    "utc": "07:19:00",
    "flight": "2L187"
  },
  {
    "day": 2,
    "utc": "19:31:00",
    "flight": "2L354"
  },
  ...
  {
    "day": 6,
    "utc": "03:42:00",
    "flight": "2L764"
  }
]
```

```
        }
    ],
    "distance": 770.969132858001
}
]
```

Landmark:

N1QLCopy

```
SELECT VALUE lm FROM landmark lm ORDER BY id LIMIT 1;
```

JSONCopy

```
[  
  {  
    "title": "Abbeville",  
    "name": "Chez Mel",  
    "alt": null,  
    "address": "63-65 rue Saint-Vulfran",  
    "directions": null,  
    "phone": "+33 3 22 19 48 64",  
    "tollfree": null,  
    "email": null,  
    "url": null,  
    "hours": null,  
    "image": null,  
    "price": null,  
    "content": "With an old style setting and musical  
accompaniment, this is a hearty and family-friendly crêpe  
restaurant. It is also a tea room in the afternoon.",  
    "geo": {  
      "lat": 50.104437,  
      "lon": 1.829432,  
    }  
  }  
]
```

```
    "accuracy": "RANGE_INTERPOLATED"
},
"activity": "eat",
"type": "landmark",
"id": 33,
"country": "France",
"city": "Abbeville",
"state": "Picardie"
}
]
```

Hotel:

N1QLCopy

```
SELECT VALUE ht FROM hotel ht ORDER BY name LIMIT 1;
```

JSONCopy

```
[  
  {  
    "title": "Avignon",  
    "name": "'La Mirande Hotel",  
    "address": "4 place de la Mirande,F- AVIGNON",  
    "directions": null,  
    "phone": null,  
    "tollfree": null,  
    "email": null,  
    "fax": null,  
    "url": null,  
    "checkin": null,  
    "checkout": null,  
    "price": "€400 and up",  
    "geo": {  
      "lat": 43.95007659797408,  
      "lon": 4.8076558113098145,  
      "accuracy": "APPROXIMATE"  
    },  
    "type": "hotel",  
  }]
```

```
"id": 1364,
"country": "France",
"city": "Avignon",
"state": "Provence-Alpes-Côte d'Azur",
"reviews": [
  {
    "content": "We stayed in this hotel for 4 nights in June 2009. After a long flight to Istanbul, we were extremely tired. When we arrived at the hotel, we were pleasantly surprised at the quality of the hotel. It was very clean and luxurious. Bathroom was large. The staff were very friendly and helpful. I would highly recommended this hotel to any one looking for a 4 to 5 star accommodation at a reasonable price. The location is very close all old town attractions.",
    "ratings": {
      "Service": 5,
      "Cleanliness": 5,
      "Overall": 5,
      "Value": 5,
      "Location": 5,
      "Rooms": 5
    },
    "author": "Marianne Wintheiser",
    "date": "2012-06-08 20:18:06 +0300"
```

```
},  
{
```

"content": "A hotel suite is normally means a connected series of rooms to be used together and not a single room as was provided. I booked what was described as Suite Executive Room and expected what I would normally get from a suite: a separate bedroom and other living area. I wanted a larger room because I was staying 8 days. I got s single room - clearly not a suite - at what was really an outrageous price. This really is misleading advertising and not acceptable. The street where the hotel is located has many other hotels and hostels as well as bars and restaurants. It is fairly downmarket with any walk down the street resulting in being accosted by touts for each. The hotel has a basic restaurant serving uninspiring food. The breakfast buffet is limited and poor. There is a very small range of breakfast cereals and fruit and some poor cold meat products that are really only suitable for feeding to the many cats in the area. There is a top floor open-air balcony but, unlike other hotels, this has no awnings or canopies and so is quite unusable in the hot afternoons. There is a separate smaller balcony area off the fourth floor but this is unfinished. There was building work and associated noise that went on until around 8:00 pm some days but this was not mentioned anywhere on the hotelâ€™s Web site. And on a final note; rooms were typically not made-up until after 4:00 pm. When we arrived back one day after 4:00 pm we

were told our room was not cleaned and were offered a drink while we waited. These drinks appeared on our bill at check-out. I have to say that my experience with this hotel is so completely at odds with that of others that I feel something is fundamentally wrong. I selected the hotel based on the Tripadvisor reviews and, based on this experience, have a significantly reduced trust in reviews on this site. The very poor value for money and overall experience invalidates any minor saving graces such as cleanliness.",

```
"ratings": {  
    "Service": 1,  
    "Cleanliness": 4,  
    "Overall": 1,  
    "Value": 1,  
    "Location": 2,  
    "Rooms": 1  
},  
"author": "Deondre Predovic III",  
"date": "2013-04-12 20:40:47 +0300"  
}  
],  
"public_likes": [  
    "Ms. Saige Hauck",  
    "Mr. Letha Lemke",  
    "Queen Farrell"
```

```
],
  "vacancy": true,
  "description": "5 star hotel housed in a 700 year old converted
townhouse",
  "alias": null,
  "pets_ok": true,
  "free_breakfast": true,
  "free_internet": true,
  "free_parking": false
}
]
```

Congratulations! You now have your Couchbase Server travel-related data being shadowed in Analytics. You're ready to start running ad hoc queries against your Analytics travel collections.

Let's go ahead and write some queries and start learning N1QL for Analytics through examples.

For your first query, let's find a particular airline based on its Couchbase Server key. You can do this for 40-Mile Air as follows:

As in SQL, the query's *FROM* clause binds the variable `a1` incrementally to the data instances residing in the Analytics collection named `airline`. Its *WHERE* clause selects only those bindings having the primary key of interest; the key is accessed (as in N1QL) by using the `meta` function to get to the meta-information about the objects.

The *SELECT* clause returns all of the meta-information plus the data value (the selected airline object in this case) for each binding that satisfies the predicate.

The expected result for this query is as follows:

JSONCopy

```
[ {  
    "meta": {  
        "id": "airline_10",  
        "vbid": 41,  
        "seq": 599,  
        "cas": 1621451375518482432,  
        "flags": 0  
    },  
    "data": {  
        "id": 10,  
        "type": "airline",  
        "name": "40-Mile Air",  
        "iata": "Q5",  
        "icao": "MLA",  
        "callsign": "MILE-AIR",  
        "country": "United States"  
    }  
}
```

```
    }  
} ]
```

Notice how the resulting object of interest has two fields whose names were requested in the *SELECT* clause.

The N1QL for Analytics language, like SQL, supports a variety of different predicates. For the next query, let's find the same airline information but in a slightly simpler or cleaner way based only on the data:

N1QLCopy

```
SELECT VALUE al  
FROM airline al  
WHERE al.name = '40-Mile Air';
```

This query's expected result is:

JSONCopy

```
[  
  {  
    "id": 10,  
    "type": "airline",  
    "name": "40-Mile Air",  
    "iata": "Q5",  
    "icao": "MLA",  
    "callsign": "MILE-AIR",  
    "country": "United States"  
  }]
```

]

In N1QL for Analytics you can select a single value (whether it be an atomic or scalar value or an object value or an array value) by using a *SELECT VALUE* clause as shown above. If you instead use the more SQL-familiar *SELECT* clause, N1QL for Analytics will return objects instead of values, and you will get a slightly differently shaped result using *SELECT al* instead of *SELECT VALUE al*:

JSONCopy

```
[  
  {  
    "al": {  
      "id": 10,  
      "type": "airline",  
      "name": "40-Mile Air",  
      "iata": "Q5",  
      "icao": "MLA",  
      "callsign": "MILE-AIR",  
      "country": "United States"  
    }  
  }  
]
```

N1QL for Analytics can apply ranges and other conditions on any data type that supports the appropriate set of comparators

```
SELECT VALUE ap
FROM airport ap
WHERE ap.geo.alt > 5000
    AND ap.airportname LIKE '%Intl%'
ORDER BY ap.name;
```

```
SELECT al.name AS airline, rt.sourceairport AS origin,
       rt.destinationairport AS destination
FROM airline al, route rt
WHERE rt.airlineid = meta(al).id
    AND rt.stops = 0
ORDER BY airline, origin, destination
LIMIT 3;
```

```
SELECT ht.city, ht.state, COUNT(*) AS num_hotels
FROM hotel ht
GROUP BY ht.city, ht.state
HAVING COUNT(*) > 30;
```

## Only on Couchbase 6.5

The following pair of Analytics DDL statements shows you how to tell Analytics to pay attention to the **beer-sample** bucket in Couchbase Server, where all of the beer and brewery information resides; they direct Analytics to shadow the data using two datasets, **breweries** and **beers**.

You need to ensure that beer-sample is loaded before proceeding ahead with this lab.

	name ▾	items	resident	ops/sec	RAM used/quota	disk used	
	beer-sample	7,303	100%	0	27MB / 100MB	34.2MB	<a href="#">Documents</a> <a href="#">Statistics</a>
	default	0	100%	0	9.7MB / 100MB	4.06MB	<a href="#">Documents</a> <a href="#">Statistics</a>

CREATE DATASET breweries ON `beer-sample` WHERE `type` = "brewery";

CREATE DATASET beers ON `beer-sample` WHERE `type` = "beer";

The screenshot shows the Couchbase Analytics Query Editor interface. It contains two separate query sections, each with a code editor, execution status, and various navigation and configuration options.

**Top Section:**

- Analytics Query Editor:** History (2/2)
- Code:**

```
1 CREATE DATASET breweries ON `beer-sample` WHERE `type` = "brewery";
```
- Status:** ✓ success | elapsed: 197.43ms | execution: 195.44ms | processed objects: 0
- Buttons:** Execute (highlighted), Explain, preferences

**Bottom Section:**

- Analytics Query Editor:** History (3/3)
- Code:**

```
1 CREATE DATASET beers ON `beer-sample` WHERE `type` = "beer";
```
- Status:** ✓ success | elapsed: 109.28ms | execution: 107.34ms | processed objects: 0
- Buttons:** Execute, Explain, preferences

**Bottom Navigation:**

- Analytics Query Results
- JSON (highlighted), Table, Tree, Plan, Plan Text

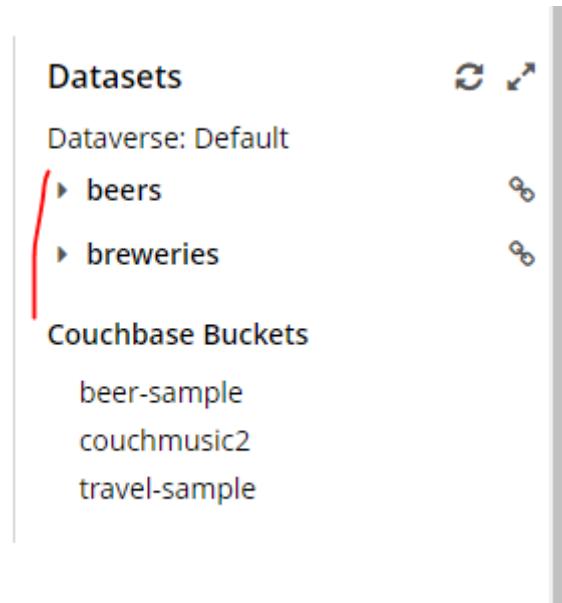
These two *CREATE DATASET* statements create the target datasets in Analytics for the information of interest. Notice how *WHERE* clauses are utilized to direct beer-related data into separate, type-specific datasets for easier querying. Each of these datasets will be hash-partitioned (sharded) across all of the nodes running instances of the Analytics service. Hash partitioning sets the stage for the parallel processing that Analytics employs when processing analytical queries.

To actually initiate the shadowing relationship of these datasets to the data in Couchbase Server, one more step is needed, namely:

```
CONNECT LINK Local;
```

Once you run this statement, Analytics begins making its copy of the Couchbase Server documents and continuously monitors it for changes. The *CONNECT LINK* statement activates the connectivity between Analytics and a Couchbase Server cluster instance. In this case, the data of interest is in the local server (i.e., it is managed by the Data service in the same cluster).

At this state, you should have dataverse as shown below:



## What's Lurking in the Shadows?

Next, verify that your datasets are really there and being populated. The following *SELECT* statements are one good way to accomplish that:

```
SELECT ds.BucketName, ds.DatasetName, ds.`Filter`  
FROM Metadata.`Dataset` ds  
WHERE ds.DataverseName = "Default";
```

The query looks in the Analytics system catalogs for datasets that have been created in the *Default* dataverse. At this point, assuming that you are just getting started, you will see only your two new datasets:

Analytics Query Editor ← history (5/5) →

```
1 SELECT ds.BucketName, ds.DatasetName, ds.`Filter`
2 FROM Metadata.`Dataset` ds
3 WHERE ds.DataVERSEName = "Default";|
```

**Execute** **Explain** ✓ success | elapsed: 97.89ms | execution: 90.43ms | count: 2 | size: 181  
| processed objects: 2    preferences

Analytics Query Results JSON Table Tree Plan Plan Text

```
1 [
2   {
3     "DatasetName": "beers",
4     "BucketName": "beer-sample",
5     "Filter": "'type' = \"beer\""
6   },
7   [
8     {
9       "DatasetName": "breweries",
10      "BucketName": "beer-sample",
11      "Filter": "type = \"brewery\""
12    }
13  ]
```

The following query asks Analytics the number of breweries:

**SELECT VALUE COUNT(\*) FROM breweries;**

It returns:

[

1412

]

The next query retrieves a sample of breweries:

```
SELECT * FROM breweries ORDER BY name LIMIT 1;
```

It returns:

```
[  
 {  
   "breweries": {  
     "address": [  
       "407 Radam, F200"  
     ],  
     "city": "Austin",  
     "code": "78745",  
     "country": "United States",  
     "description": "(512) Brewing Company is a microbrewery located in the heart of Austin  
that brews for the community using as many local, domestic and organic ingredients as  
possible.",  
     "geo": {  
       "accuracy": "ROOFTOP",  
       "lat": 30.2234,  
       "lon": -97.7697  
     },  
     "name": "(512) Brewing Company",  
     "phone": "512.707.2337",  
     "state": "Texas",  
   }  
 }]
```

```
"type": "brewery",
"updated": "2010-07-22 20:00:20",
"website": "http://512brewing.com/"
}
]
```

The following query asks Analytics the number of beers:

```
SELECT VALUE COUNT(*) FROM beers;
```

It returns:

```
[  
 5891  
]
```

The following query retrieves a sample of beers:

```
SELECT * FROM beers ORDER BY name LIMIT 1;
```

It returns:

```
[  
{  
  "beers": {  
    "abv": 0,
```

```
"brewery_id": "big_ridge_brewing",
"category": "North American Lager",
"description": "",
"ibu": 0,
"name": "#17 Cream Ale",
"srm": 0,
"style": "American-Style Lager",
"type": "beer",
"upc": 0,
"updated": "2010-07-22 20:00:20"
}
]
}
```

### Query 0 - Key-Based Lookup

For your first query, let's find a particular brewery based on its Couchbase Server key. You can do this for the Kona Brewing company as follows:

```
SELECT meta(bw) AS meta, bw AS data
FROM breweries bw
WHERE meta(bw).id = 'kona_brewing';
```

As in SQL, the query's *FROM* clause binds the variable **bw** incrementally to the data instances residing in the dataset named *breweries*. Its *WHERE* clause selects only those bindings having the primary key of interest; the key is accessed (as in N1QL) by using the

meta function to get to the meta-information about the objects. The *SELECT* clause returns all of the meta-information plus the data value (the selected brewery object in this case) for each binding that satisfies the predicate.

The expected result for this query is as follows:

Analytics Query Editor ← history (10/10) →

```
1 SELECT meta(bw) AS meta, bw AS data
2 FROM breweries bw
3 WHERE meta(bw).id = 'kona_brewing';
```

**Execute** **Explain** ✓ success | elapsed: 96.39ms | execution: 89.44ms | count: 1 | size: 491 ⚙️ preferences

Analytics Query Results 

**JSON** Table Tree Plan Plan Text

```
1+ [{"2+   "meta": {3+     "id": "kona_brewing",4+     "vbid": 862,5+     "seq": 5,6+     "cas": 1556859604414627840,7+     "flags": 335544328+   },9+   "data": {10+     "address": ["75-5629 Kuakini Highway"],11+     "city": "Kailua-Kona",12+     "code": "96740",13+     "country": "United States",14+     "description": "",15+     "geo": {}}
```

```
15      "geo": {  
16          "accuracy": "RANGE_INTERPOLATED",  
17          "lat": 19.642,  
18          "lon": -155.996  
19      },  
20      "name": "Kona Brewing",  
21      "phone": "1-808-334-1133",  
22      "state": "Hawaii",  
23      "type": "brewery",  
24      "updated": "2010-07-22 20:00:20",  
25      "website": "http://www.konabrewingco.com"  
26  }  
27 ]
```

Notice how the resulting object of interest has two fields whose names were requested in the SELECT clause.

**JOIN** - You wanted a list of all breweries paired with their associated beers, with the list enumerating the brewery name and the beer name for each such pair. You can do this as follows in N1QL for Analytics while also limiting the answer set size to at most 3 results:

```
SELECT bw.name AS brewer, br.name AS beer  
FROM breweries bw, beers br  
WHERE br.brewery_id = meta(bw).id  
ORDER BY bw.name, br.name  
LIMIT 3;
```

The screenshot shows the Couchbase Analytics Query Editor interface. At the top, there's a code editor window containing the following SQL-like query:

```

1 SELECT bw.name AS brewer, br.name AS beer
2 FROM breweries bw, beers br
3 WHERE br.brewery_id = meta(bw).id
4 ORDER BY bw.name, br.name
5 LIMIT 3;

```

Below the code editor are two buttons: "Execute" (highlighted in blue) and "Explain". To the right of these buttons, the status bar shows "success" with an elapsed time of 322.91ms, an execution time of 309.78ms, a count of 3, a size of 179, and processed objects of 7303. There's also a "preferences" link.

Below the status bar is the "Analytics Query Results" section. It has a "JSON" tab (which is selected), a "Table" tab, a "Tree" tab, a "Plan" tab, and a "Plan Text" tab. The JSON results pane displays the following data:

```

1 [
2   {
3     "brewer": "(512) Brewing Company",
4     "beer": "(512) ALT"
5   },
6   {
7     "brewer": "(512) Brewing Company",
8     "beer": "(512) Bruin"
9   },
10  {
11    "brewer": "(512) Brewing Company",
12    "beer": "(512) IPA"
}

```

## Grouping and Aggregation

For each brewery that offers more than 30 beers, the following group-by or aggregate query reports the number of beers that it offers.

```

SELECT br.brewery_id, COUNT(*) AS num_beers
FROM beers br
GROUP BY br.brewery_id
HAVING COUNT(*) > 30

```

ORDER BY COUNT(\*) DESC;

Analytics Query Editor

```
1 SELECT br.brewery_id, COUNT(*) AS num_beers
2 FROM beers br
3 GROUP BY br.brewery_id
4 HAVING COUNT(*) > 30
5 ORDER BY COUNT(*) DESC;
```

history (18/18)

Execute Explain ✓ success | elapsed: 254.49ms | execution: 243.17ms | count: 11 | size: 607 | processed objects: 5891 preferences

Analytics Query Results

JSON Table Tree Plan Plan Text

```
1 [
2   {
3     "num_beers": 57,
4     "brewery_id": "midnight_sun_brewing_co"
5   },
6   {
7     "num_beers": 49,
8     "brewery_id": "rogue_ales"
9   },
10  {
11    "num_beers": 38,
12    "brewery_id": "anheuser_busch"
13  }
14]
```

----- Lab ends Here -----

## 25. Monitoring and Troubleshooting – 60 Minutes

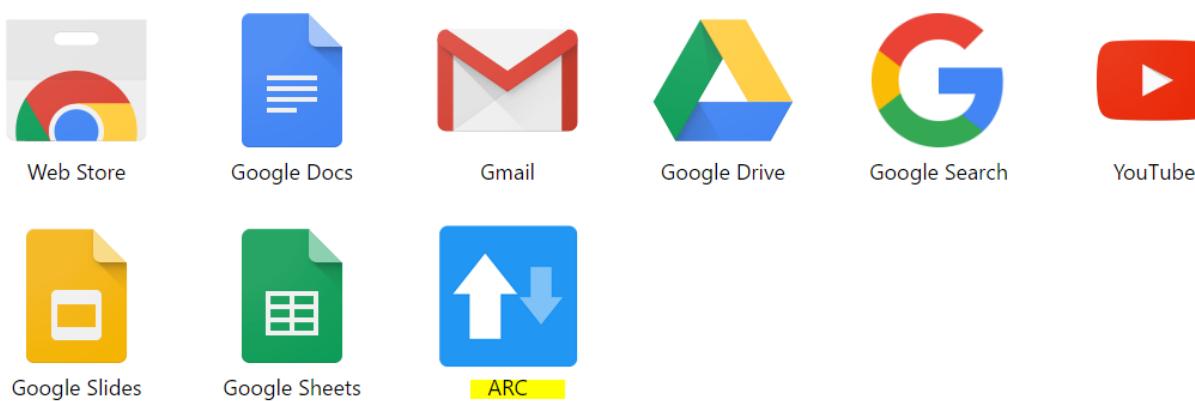
In this lab we will understand the following:

- Using REST API Client to monitor Couchbase cluster
- Bucket statistics Using Cbstats
- Rebalance status
- View vBucket Details Using CB Dump

Install Advance rest Client for Chrome using the following:

<https://chrome.google.com/webstore/category/extensions>

Open ARC Using Apps Icon of Chrome



Double click on ARC.

Supply the request URL as shown below:

<http://tos.hp.com:8091/pools/default/buckets/beer-sample/stats>

Statistics can be retrieved at the bucket level. The request URL should be taken from stats.uri property of a bucket response. By default, this request returns stats for the last minute and for heavily used keys. Query parameters provide a more detailed level of information.

The screenshot shows a REST client interface with the following details:

- Request URL:** http://192.168.188.165:8091/pools/default/buckets/beer-samples/stats
- Method:** GET (selected)
- Headers:** Content-Type: application/json
- Buttons:** Use XHR, SEND

Click on Send

Supply the credentials.[Administrator/admin123] in Authorization section

## Authentication required

The endpoint requires a username and password

login  
Administrator

---

password

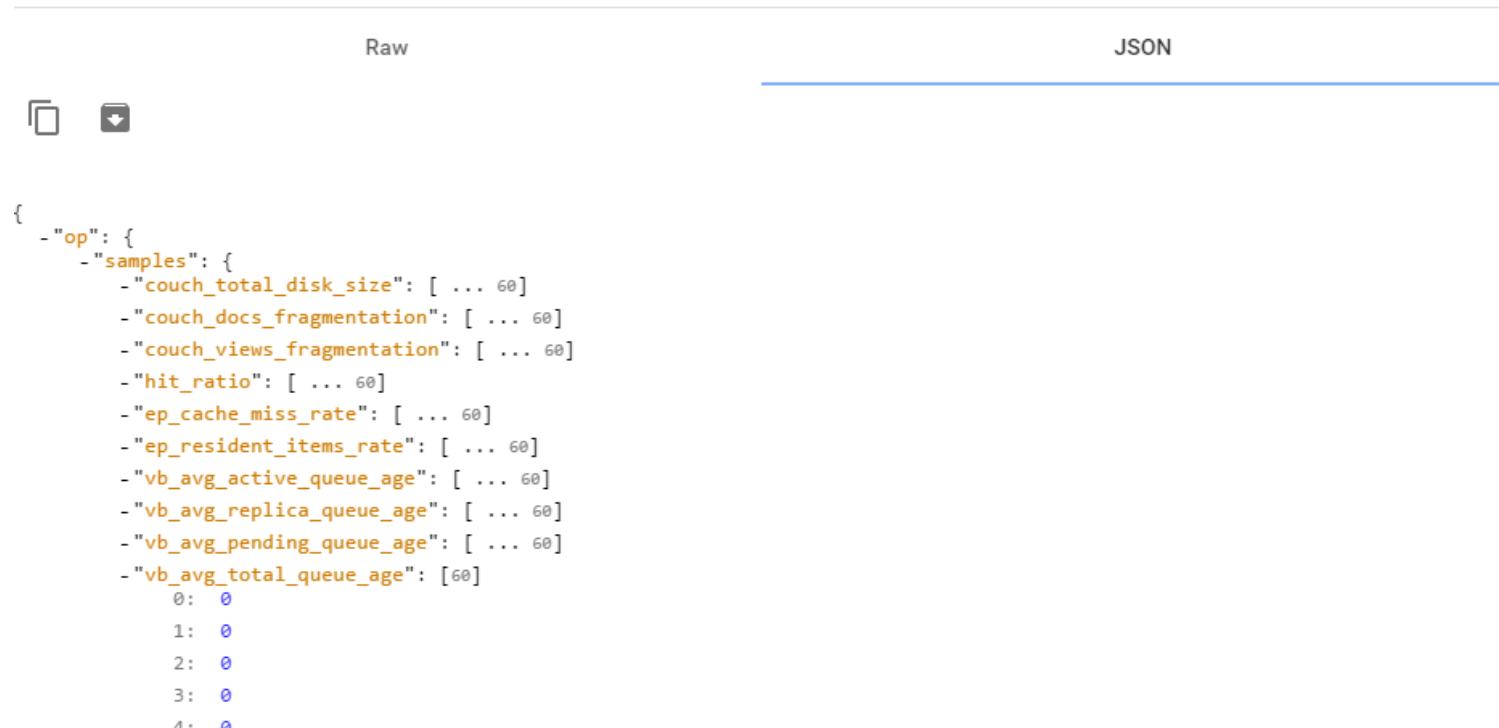
.....| 

---

[CANCEL](#) [LOG IN](#)

---

You will get the result:



```
{
  - "op": {
    - "samples": {
      - "couch_total_disk_size": [ ... 60]
      - "couch_docs_fragmentation": [ ... 60]
      - "couch_views_fragmentation": [ ... 60]
      - "hit_ratio": [ ... 60]
      - "ep_cache_miss_rate": [ ... 60]
      - "ep_resident_items_rate": [ ... 60]
      - "vb_avg_active_queue_age": [ ... 60]
      - "vb_avg_replica_queue_age": [ ... 60]
      - "vb_avg_pending_queue_age": [ ... 60]
      - "vb_avg_total_queue_age": [60]
        0: 0
        1: 0
        2: 0
        3: 0
        4: 0
        ...
      ]
    }
  }
}
```

**Verify the: Couch total disk size; Shown as byte**

```
- "op": {
  - "samples": {
    - "couch_total_disk_size": [Array[60]
      0: 94200571
      1: 94200571,
      2: 94200571,
```

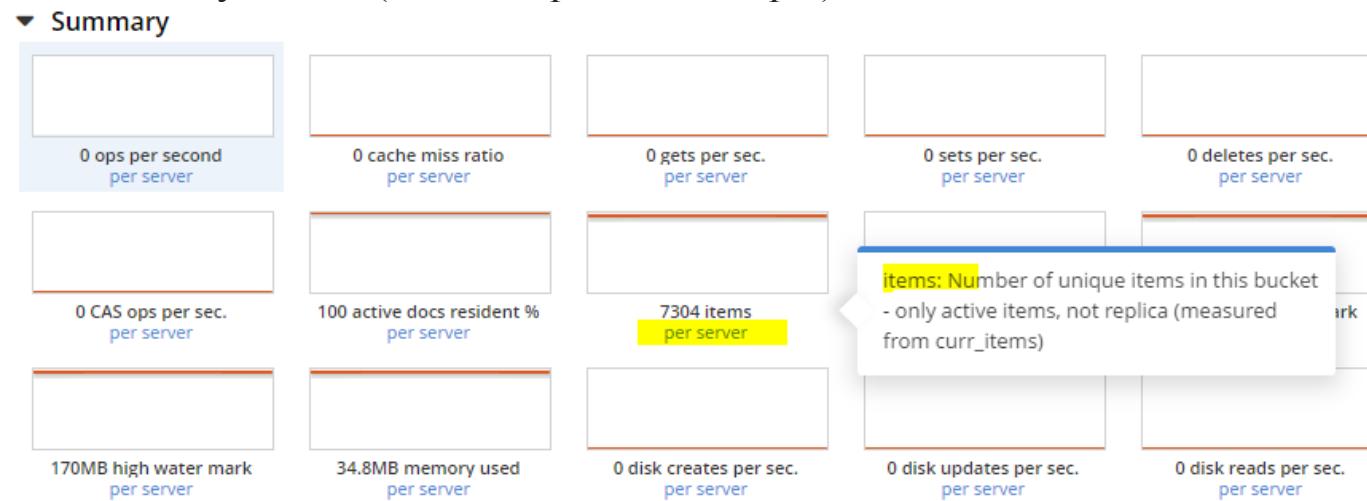
In our example its approx. to 89 MB

**You can use the json viewer to view the json**

<http://jsonviewer.stack.hu/>

You can use the web console to view the statistics. Buckets → Beer-sample → Statistics

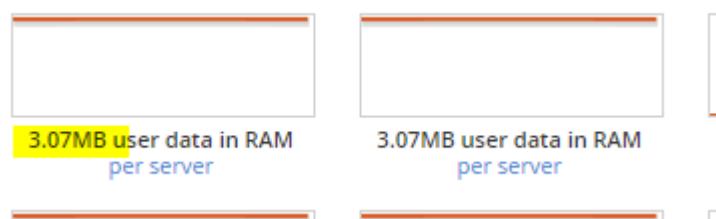
Click on Any bucket (For example beer-sample).

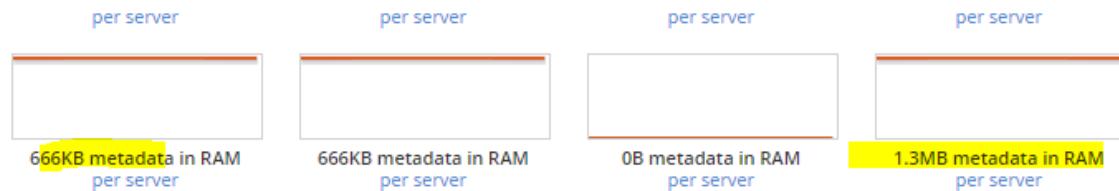


Hover your mouse on the counter to determine the purpose of the counter. Here, items displayed the number of documents which are active in the beer-sample.

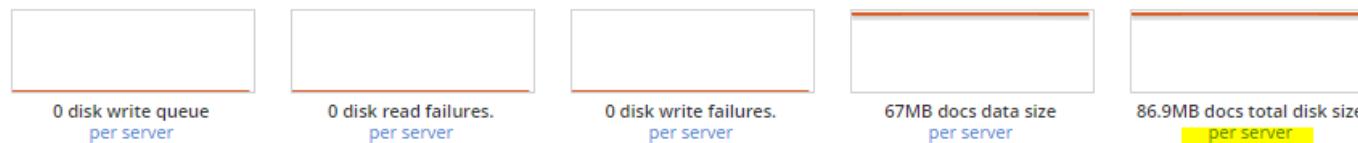
Try to find out how much RAM is use by the meta and data.

## vBucket Resources - Section





It display for active and replica vBucket.  
Summary → Total Disk Size consumed:



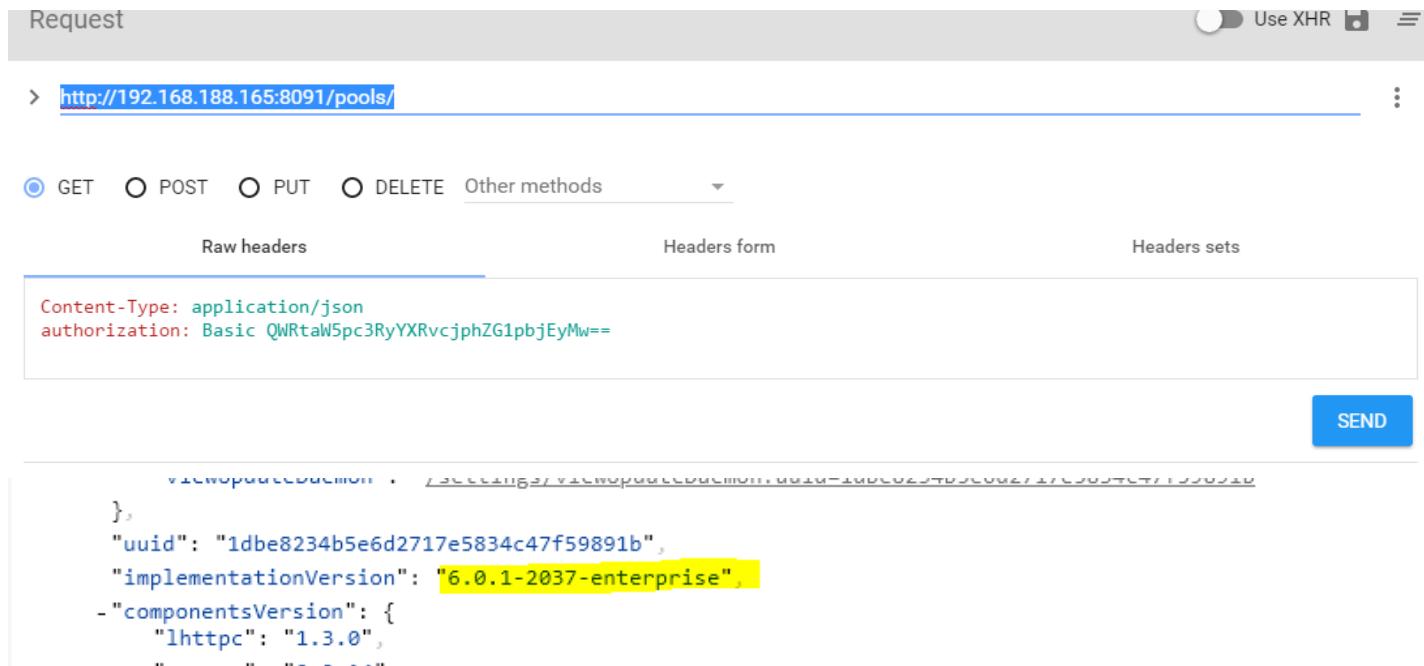
## Monitoring cluster tasks

You can use the REST API or Couchbase Server command-line tools to monitor various cluster tasks, such as node warmup, rebalance operations, and more. In this section examples for retrieving the following per-node operational information from a Couchbase Server cluster will be detailed to familiarize you with the resources available to you.

Example operational information:

- Retrieves cluster information.

<http://tos.hp.com:8091/pools/>



shows the version of CB.

- Node warmup status using CLI . Go to the Couchbase installation folder // User Credentials is required for executing the mention command.

```
./cbstats -b beer-sample tos.hp.com:11210 -u Administrator -p admin123 all | grep warmup
```

```
[root@tos bin]# ./cbstats -b beer-sample tos.hp.com:11210 -u Administrator -p admin123 all | grep warmup
ep_failpartialwarmup: false
ep_waitforwarmup: false
ep_warmup: true
ep_warmup_batch_size: 10000
ep_warmup_dups: 0
ep_warmup_min_items_threshold: 100
ep_warmup_min_memory_threshold: 100
ep_warmup_oom: 0
ep_warmup_thread: complete
ep_warmup_time: 105369
[root@tos bin]#
```

You can use the cbstats utility on the node in question to quickly monitor warmup. Here is an example with output showing warmup in complete (note ep\_warmup\_thread: complete and how long it took to warm up is provided by ep\_warmup\_time):

Low water mark for auto-evictions.

```
./cbstats tos.hp.com:11210 -b travel-sample all -u Administrator -p admin123 | grep
ep_mem_low_wat
```

```
[root@tos bin]# ./cbstats tos.hp.com:11210 -b beer-sample all -u Administrator -p admin123 | grep ep_mem_low_wat
ep_mem_low_wat: 78643200
ep_mem_low_wat_percent: 0.75
```

75 MB is the low water mark to stop the eviction.

- Rebalance progress

```
# curl --silent -u Administrator:admin123 tos.hp.com:8091/pools/default/tasks \
| python -mjson.tool | grep -A1 -B4 rebalance
```

```
[root@tos bin]# curl --silent -u Administrator:admin123 localhost:8091/pools/default/tasks \
> | python -mjson.tool | grep -A1 -B4 rebalance
{
    "masterRequestTimedOut": false,
    "status": "notRunning",
    "statusIsStale": false,
    "type": "rebalance"
}
[root@tos bin]#
```

Use curl, python, and grep with the REST API to check rebalance status:

From this simplified example, you can see that rebalance is not running. If you omit the grep command, you can get detailed rebalance progress metrics as well.

- Let us execute cbstats command for a particular and understand some of the few parameters.

```
./cbstats tos.hp.com:11210 -b beer-sample all -u Administrator -p admin123 > /tmp/all.txt
```

```
wbufs_loaned: 74039
[root@tos bin]# ./cbstats tos.hp.com:11210 -b beer-sample all -u Administrator -p admin123 > /tmp/all.txt
[root@tos bin]#
```

Open the file using vi tools and verify it as shown below, to familiarize it.

- By tracking the number of open and rejected connections via `curr_connections` and `rejected_conns` statistics, we can understand if any connection requests were rejected by this node.
- Each time an object is requested by an application and not found in the cache, Couchbase Server will find the object on disk.

This cache miss requires a background fetch and is measured per item fetched from disk by `ep_bg_fetched`. If you are managing to a 100% working set, this could be a sign of a cluster under stress. The cache miss might not be an issue if you have a smaller working set. In either scenario, gaining an understanding of what's typical in an environment is important as a large increase will provide an early warning signal.

3. The number of items queued for persistence is an important area to monitor so that you understand if your cluster has adequate I/O resources for your application.

While your application will always be served via the caching tier, one of the great benefits of Couchbase Server is its ability to provide data durability by persisting to disk. If this asynchronous operation becomes overloaded, it can impact the application performance. As a result, especially in heavy write systems, it will be important that you monitor `ep_queue_size` and `ep_flusher_todo`. You would never want to get to 1 million items in the queue, and you would likely want to flag a warning around 500000 to 800000, especially if this is an upward trend over time.

4. Following the `vb_num_eject_replicas` statistic gives you the number of times when Couchbase Server ejected replica values from memory.

This statistic indicates that a specific bucket has reached its low watermark. While simply reaching this threshold might not be problematic as the cluster is simply freeing memory resources, it might be a problem if you consistently see this behavior or if it is an increasing trend. More importantly, this is a way to head off out-of-memory (`ep_oom_errors` and `ep_tmp_oom_errors`) scenarios, which you never want to see in your production clusters.

If there is any issue in a particular vbUcket. You can diagnosis that bucket using the following steps:  
Identify a single vbucket that exhibits the problem

- Immediately narrows the problem space to 1/1024th of the data set
- `cbstats vbucket-details`
- Interrogate the files on disk to find the discrepancy
- `couch_dbdump --no-body --json --by-id <file>`
- Diff the source cluster and the destination cluster

```
#cbstats tos.hp.com:11210 -u Administrator -p admin123 -b beer-sample vbucket-details 1022
```

```
C:\Program Files\Couchbase\Server\bin>cbstats 192.168.139.136:11210 -u Administrator -p admin123 -b beer-sample vbuckets  
-details 1022  
vb_1022:active  
vb_1022:backfill_queue_size:  
vb_1022:bloom_filter:  
vb_1022:bloom_filter_key_count:  
vb_1022:bloom_filter_size:  
vb_1022:db_data_size:  
vb_1022:db_file_size:  
vb_1022:drift_ahead_threshold:  
vb_1022:drift_ahead_threshold_exceeded:  
vb_1022:drift_behind_threshold:  
vb_1022:drift_behind_threshold_exceeded:  
vb_1022:high_seqno:  
vb_1022:hp_vb_req_size:  
vb_1022:ht_cache_size:  
vb_1022:ht_item_memory:  
vb_1022:ht_item_memory_uncompressed:  
vb_1022:ht_memory:  
vb_1022:ht_size:  
vb_1022:logical_clock_ticks:  
vb_1022:max_cas:  
vb_1022:max_cas_str:  
vb_1022:max_deleted_revid:  
vb_1022:might_contain_xattrs:  
vb_1022:num_ejects:  
vb_1022:num_items:  
vb_1022:num_non_resident:  
vb_1022:num_temp_items:  
11
```

In this case, the vbucket 1022 has 11 items in it.

Interrogate the files on disk to find the discrepancy:

```
#cd /opt/couchbase/var/lib/couchbase/data/beer-sample  
#/opt/couchbase/bin/couch_dbdump 1022.couch.l
```

```
(base) [root@tos beer-sample]# /opt/couchbase/bin/couch_dbdump 1022.couch.1
Dumping "1022.couch.1":
Doc seq: 1
  id: staatliches_hofbrauhaus_in_munchen-hofbrau_oktoberfestbier
  rev: 1
  content_meta: 128
  size (on disk): 498
  cas: 1574233347993305088, expiry: 0, flags: 33554432, datatype: 0x01 (json)
  size: 535
  data: (snappy) {"abv":6.3,"brewery_id":"staatliches_hofbrauhaus_in_munchen","category":"German Lager","description":"Hofbräu brews a rich, full-bodied beer which goes down ideally with traditional Bavarian cuisine. With its deliciously bitter taste and alcoholic content of 6.3% volume, Hofbräu Oktoberfestbier is as special as the Beer Festival itself.\r\n\r\nType: Bottom-fermented, light festive beer","ibu":0.0,"name":"Hofbräu Oktoberfestbier","srm":0.0,"style":"German-Style Oktoberfest","type":"beer","upc":0,"updated":"2010-07-22 20:00:20"}
Doc seq: 2
  id: sierra_nevada_brewing_co-harvest_ale_2007
  rev: 1
  content_meta: 128
  size (on disk): 228
  cas: 1574233348071227392, expiry: 0, flags: 33554432, datatype: 0x01 (json)
  size: 234
  data: (snappy) {"abv":6.7,"brewery_id":"sierra_nevada_brewing_co","category":"North American Ale","description":"","ibu":0.0,"name":"Harvest Ale 2007","srm":0.0,"style":"American-Style Pale Ale","type":"beer","upc":0,"updated":"2010-07-22 20:00:20"}

  data: (Snappy) { abv : 5.0, brewery_id : pennichuck_brewing_company , category : British Ale , description : , ibu : 0.0, name : Bagpiper's Scottish Ale,"srm":0.0,"style":"Scotch Ale","type":"beer","upc":0,"updated":"2010-07-22 20:00:20"}
Doc seq: 11
  id: day_brewing
  rev: 1
  content_meta: 128
  size (on disk): 260
  cas: 1574233353508290560, expiry: 0, flags: 33554432, datatype: 0x01 (json)
  size: 260
  data: (snappy) {"address":[],"city":"Marrero","code":"","country":"United States","description":"","geo":{"accuracy":"APPROXIMATE","lat":29.8994,"lon":-90.1004}, "name":"Day Brewing","phone":"","state":"Louisiana","type":"brewery","updated":"2010-07-22 20:00:20","website":""}

Total docs: 11
(base) [root@tos beer-sample]#
```

As displayed above there are 11 documents in this vBucket 1022.

*Only by ID: Get all the Document Id in a particulare vBucket*

```
/opt/couchbase/bin/couch_dbdump --no-body --json --byid 1022.couch.1
```

```
(base) [root@tos beer-sample]# /opt/couchbase/bin/couch_dbdump --no-body --json --byid 1022.couch.1
Dumping "1022.couch.1":
{"seq":3,"id":"appleton_brewing","rev":1,"content_meta":128,"physical_size":258,"cas":"1574233348118544384","expiry":0,"flags":33554432,"datatype":1,"datatype_as_text":["json"],"body":null}
 {"seq":5,"id":"beach_brewing-harvest_gold","rev":1,"content_meta":128,"physical_size":154,"cas":"1574233348197515264","expiry":0,"flags":33554432,"datatype":1,"datatype_as_text":["json"],"body":null}
 {"seq":9,"id":"bootleggers_stakehouse_and_brewery-double_zz_raspberry_wheat","rev":1,"content_meta":128,"physical_size":250,"cas":"157423352737325056","expiry":0,"flags":33554432,"datatype":1,"datatype_as_text":["json"],"body":null}
 {"seq":6,"id":"brouwerij_kerkom-bloesem_bink","rev":1,"content_meta":128,"physical_size":163,"cas":"1574233348836425728","expiry":0,"flags":33554432,"datatype":1,"datatype_as_text":["json"],"body":null}
 {"seq":11,"id":"day_brewing","rev":1,"content_meta":128,"physical_size":260,"cas":"1574233353508290560","expiry":0,"flags":33554432,"datatype":1,"datatype_as_text":["json"],"body":null}
 {"seq":4,"id":"jj_bitting_brewing-hop_garden_pale_ale","rev":1,"content_meta":128,"physical_size":221,"cas":"1574233348170514432","expiry":0,"flags":33554432,"datatype":1,"datatype_as_text":["json"],"body":null}
 {"seq":8,"id":"otto_s_pub_and_brewery-red_mo_ale","rev":1,"content_meta":128,"physical_size":354,"cas":"1574233351340621824","expiry":0,"flags":33554432,"datatype":1,"datatype_as_text":["json"],"body":null}
 {"seq":10,"id":"pennichuck_brewing_company-bagpiper_s_scottish_ale","rev":1,"content_meta":128,"physical_size":219,"cas":"1574233353350283264","expiry":0,"flags":33554432,"datatype":1,"datatype_as_text":["json"],"body":null}
 {"seq":7,"id":"rogue_ales-morimoto_imperial_pilsner","rev":1,"content_meta":128,"physical_size":735,"cas":"1574233348930797568","expiry":0,"flags":33554432,"datatype":1,"datatype_as_text":["json"],"body":null}
 {"seq":2,"id":"sierra_nevada_brewing_co-harvest_ale_2007","rev":1,"content_meta":128,"physical_size":228,"cas":"1574233348071227392","expiry":0,"flags":33554432,"datatype":1,"datatype_as_text":["json"],"body":null}
 {"seq":1,"id":"staatliches_hofbrauhaus_in_munchen-hofbrau_oktoberfestbier","rev":1,"content_meta":128,"physical_size":498,"cas":"157423347993305088","expiry":0,"flags":33554432,"datatype":1,"datatype_as_text":["json"],"body":null}

Total docs: 11
(base) [root@tos beer-sample]#
```

Scenario : Server Itself is Down , What matters is server status at the time of the event.  
Verify the status of the server at that time, i.e verify the uptime of the server.

```
#cd /opt/couchbase/bin
./cbstats tos.hp.com:11210 -u Administrator -p admin123 -b travel-sample all | grep uptime
```

```
[root@tos bin]# ./cbstats tos.hp.com:11210 -u Administrator -p admin123 -b travel-sample all | grep uptime
uptime: 3993
[root@tos bin]# ./cbstats tos.slavea.com:11210 -u Administrator -p admin123 -b travel-sample all | grep uptime
uptime: 2126
```

Check for the node which is having issue. In the above, node hp was up for 3993 seconds.

yum install wireshark - OOptional

-----End of Lab-----

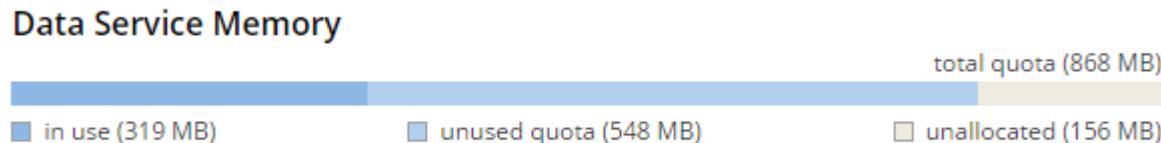
## 26. Advance Cluster Settings – 30 Minutes

In this lab, we will perform the following:

- Modification of RAM Quotas for each node.
- Index Thread setting.
- Auto failover setting
- Email Configuration
- Compaction

**Increases RAM quota for Data service in each node by 50 MB and verify it from the dashboard.**

Let us observe the current RAM quota for each node using Dashboard view.



Here, Total Allocated to the cluster = 1023 i.e sum of in use, unused quota and unallocated.  
 Total RAM allocated to buckets = 868 MB, you can view using Buckets view.

name ▾	items	resident	ops/sec	RAM used/quota
beer-sample	7,305	100%	0	27.4MB / 100MB
couchmusic2	213,063	100%	0	268MB / 512MB
default	1	100%	0	22.9MB / 256MB

Increase the Quota by 50 MB. For this go to Settings → Cluster and modify the Memory Quota for Data services.

Cluster Name (0 — 256 chars)  
Administrator

Couchbase Memory Quotas in megabytes per server node

Data Service	<input type="text" value="1024"/> MB
Index Service	<input type="text" value="512"/> MB
Search Service	<input type="text" value="512"/> MB
Query Service	-----

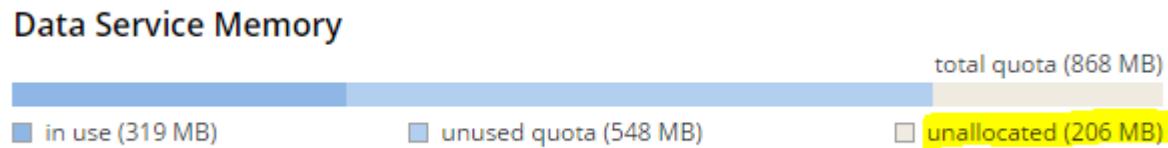
Update by +50 MB and click Save.

**Cluster Name** (0 — 256 chars)

**Couchbase Memory Quotas** in megabytes per server node

Data Service	1074	MB
Index Service	512	MB
Search Service	512	MB
Query Service	-----	

Finally we can verify actual Quota now from the Dashboard View



You can observed that unallocated size has been increased by 50MB.

**Index Thread setting.**

You can increase the Indexers thread to 2 using the Advance settings. If you have resources and require indexing frequently then you can increase from here.

The screenshot shows the Couchbase Admin UI with the 'Cluster' tab selected. On the left sidebar, the 'Logs' section is highlighted with a red box. In the main content area, under 'Advanced Index Settings', the 'Indexer Threads' input field is also highlighted with a red box and contains the value '0'. Other configuration fields include 'Index Storage Mode' (Standard Global Secondary selected), 'Service Memory Quotas' (Data, Index, Search, Query, Eventing all set to 256 MB, Analytics set to 1024 MB), and 'Indexer Log Level' (Info).

Category	Setting	Value
Service Memory Quotas	Data	256 MB
	Index	256 MB
	Search	256 MB
	Query	-----
	Eventing	256 MB
	Analytics	1024 MB
<b>Indexer Threads</b>		0
<b>Indexer Log Level</b>		Info

**Index Storage Mode**

- Standard Global Secondary  
 Memory-Optimized (i)

▼ Hide Advanced Index Settings

**Indexer Threads (i)**

2

**Max Rollback Points (i)**

5

**Indexer Log Level**

Info

Save

Click Save.

**Auto failover setting.**

The screenshot shows the Couchbase Settings interface with the 'Node Availability' tab selected. On the left, a sidebar lists various settings categories: Dashboard, Servers, Buckets, XDCR, Security, **Settings**, and Logs. The 'Settings' category is highlighted with a red underline. Below the sidebar, there are three configuration sections:

- Failover Nodes Automatically ⓘ**
  - Enable auto-failover after  seconds for up to  event
  - Enable auto-failover for sustained data disk read/write failures after  seconds
  - Enable auto-failover of server groups ⚠
- ▶ For Ephemeral Buckets

A blue 'Save' button is located at the bottom of the main content area.

## Email Setting

You can configure the SMTP setting using this option.

The screenshot shows the Couchbase Admin UI with the 'Email Alerts' tab selected. On the left, there's a sidebar with navigation links: Dashboard, Servers, Buckets, Indexes, Search, Query, XDCR, Security, Settings (which is currently selected), and Logs. The main content area has a heading 'Email Server Settings' and fields for Host (localhost), Port (25), Username, and Password.

## Schedule compaction

You can configure compaction using this option, Settings → Auto Compaction.

### Auto-compaction Settings

Auto-Compaction settings trigger the compaction process.

This process compacts databases and their respective view indexes when the following conditions are met.

#### Database Fragmentation

Set the database fragmentation level to determine the point when compaction is triggered.

20 %

MB

#### View Fragmentation

Set the view fragmentation level to determine the point when compaction is triggered.

30 %

MB

#### Time Interval

Set the time interval for when compaction is allowed to run

##### Start Time

21 : 00

##### End Time

23 : 00

Here, we are triggering the compaction when Database fragmentation cross 20 % and will be executed during 21.00 Hrs and 23.00 Hrs only.

You can go through the various options to have a feel of it as an administrator.

### Log Redaction

Log → Collect Information

To perform explicit logging with redaction by means of Couchbase Web Console, before starting the collection-process, access the Log Redaction Level panel, on the Collect Information screen. This features two radio-buttons, labelled None and Partial Redaction. Make sure the Partial Redaction radio-button is selected. Guidance on redaction is displayed below it:

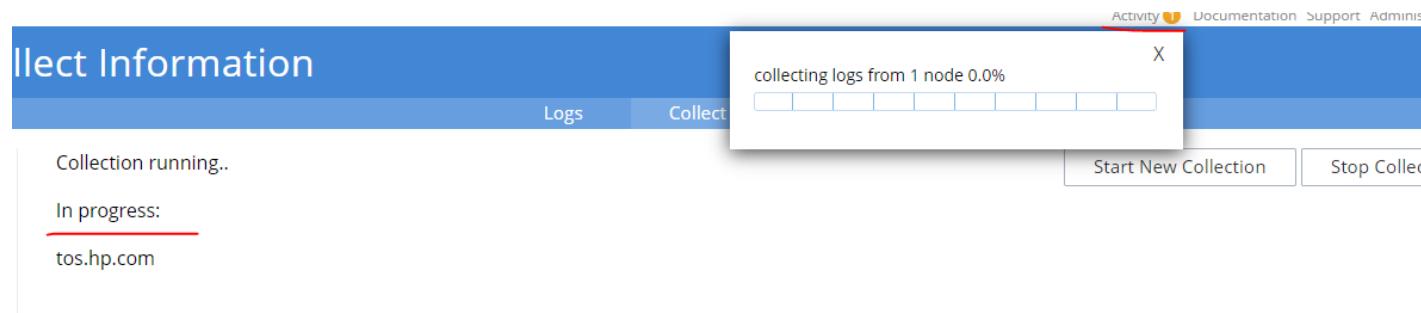
#### Log Redaction Level

None  Partial Redaction  ⓘ

Couchbase Server will collect and save a redacted log file at the location you specify, but also save an unredacted version which could be useful for further troubleshooting.

If you use the "Upload to Couchbase" feature below, ONLY the redacted log will be uploaded.

Left-click on the Start Collection button. A notification explains that the collection-process is now running. When the process has completed, a further notification appears, specifying the location (local or remote) of each created zip file. Note that, when redaction has been specified, two zip files are provided for each node: one file containing redacted data, the other unredacted data



After sometimes you will get the following result.

The screenshot shows a user interface titled 'Collect Information'. At the top, there are two tabs: 'Logs' (which is selected) and 'Collect Information'. Below the tabs, a message says 'Collection completed'. Underneath, it states 'Logs were successfully collected to the following paths:' followed by a list: 'tos.hp.com /opt/couchbase/var/lib/couchbase/tmp/collectinfo-2019-12-29T132827-ns\_1@tos.hp.com.zip'. A red line highlights the path 'tos.hp.com /opt/couchbase/var/lib/couchbase/tmp/collectinfo-2019-12-29T132827-ns\_1@tos.hp.com.zip'.

You can upload the log to couchbase server.

-----End of Lab-----

## 27. Request profiling & Query Monitoring (Optional) – 90 Minutes

Profile request REST API parameter, passed to the query service alongside the request statement – ideal for investigating individual statements.

```
curl http://localhost:8093/query/service -d "statement=select * from `travel-sample` limit 1 &profile=timings" -u Administrator:admin123
```

```
C:\Users\henryp>curl http://localhost:8093/query/service -d "statement=select * from `travel-sample` limit 1 &profile=timings" -u Administrator:admin123
{
  "requestID": "6f846645-9b8f-4f0e-acc4-d04d789d1d2c",
  "signature": {"*": "*"},
  "results": [
    {"travel-sample": {"callsign": "MILE-AIR", "country": "United States", "iata": "Q5", "icao": "MLA", "id": 10, "name": "40-Mile Air", "type": "airline"}}
  ],
  "status": "success",
  "metrics": {"elapsedTime": "16.9833ms", "executionTime": "16.9833ms", "resultCount": 1, "resultSize": 138},
  "profile": {"phaseTimes": {"fetch": "1.0006ms", "parse": "942.97ms", "primaryScan": "15.0398ms", "run": "16.0404ms"}, "phaseCounts": {"fetch": 1, "primaryScan": 1}, "phaseOperators": {"authorize": 1, "fetch": 1, "primaryScan": 1}, "executionTimings": {"#operator": "Sequence", "#stats": {"#phaseSwitches": 2, "kernTime": "16.0404ms"}, "~children": [{"#operator": "Authorize", "#stats": {"#phaseSwitches": 4, "kernTime": "16.0404ms"}}, {"privileges": {"List": [{"Target": "default:travel-sample", "Priv": 7}]}], "~child": [{"#operator": "Sequence", "#stats": {"#phaseSwitches": 3, "kernTime": "16.0404ms"}}, {"~children": [{"#operator": "PrimaryScan", "#stats": {"#itemsOut": 1, "#phaseSwitches": 7, "execTime": "15.0398ms"}, "index": "def_primary", "keyspace": "travel-sample", "limit": "1", "namespace": "default", "using": "gsi"}, {"#operator": "Fetch", "#stats": {"#itemsIn": 1, "#itemsOut": 1, "#phaseSwitches": 11, "execTime": "1.0006ms", "kernTime": "15.0398ms"}, "keyspace": "travel-sample", "namespace": "default"}, {"#operator": "Sequence", "#stats": {"#phaseSwitches": 5, "kernTime": "16.0404ms"}, "~children": [{"#operator": "InitialProject", "#stats": {"#itemsIn": 1, "#itemsOut": 1, "#phaseSwitches": 9, "kernTime": "16.0404ms"}, "result_terms": [{"expr": "self", "star": true}], "#operator": "FinalProject", "#stats": {"#itemsIn": 1, "#itemsOut": 1, "#phaseSwitches": 11, "kernTime": "16.0404ms"}}], {"#operator": "Limit", "#stats": {"#itemsIn": 1, "#itemsOut": 1, "#phaseSwitches": 11, "kernTime": "16.0404ms"}, "expr": "1"}], {"#operator": "Stream", "#stats": {"#itemsIn": 1, "#itemsOut": 1, "#phaseSwitches": 9, "kernTime": "16.0404ms"}}, {"~versions": ["2.0.0-N1QL", "5.0.1-5003-enterprise"]}]}}
```

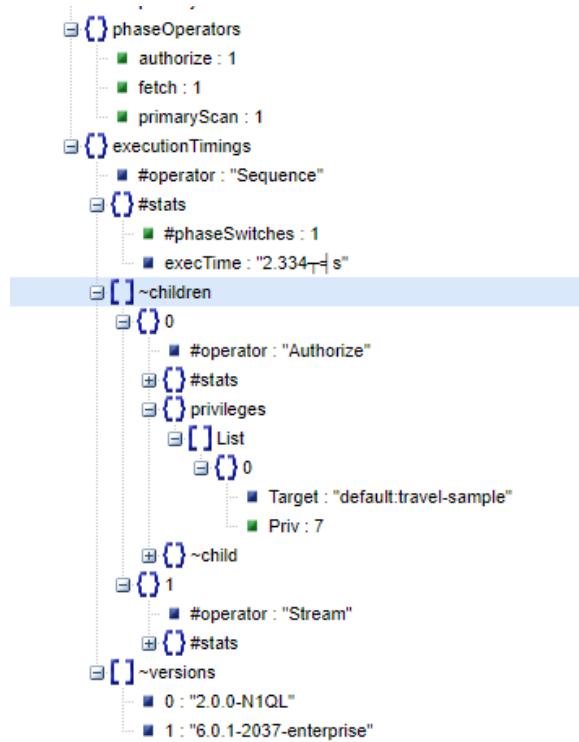
It's not readable. Hence You can paste the above Json in any JSON viewer tools as shown below for readability. We will use <http://jsonviewer.stack.hu/> for this lab.

Verify the following:

How much time it took to parse the Query: (Parse)

How much time it took to scan the index and what index being used: (Primaryscan)

```
JSON
  - requestId : "92438de0-d82c-4f80-ad78-b4b16452d9f5"
  - signature
    - *
  - results
    - 0
      - travel-sample
        - callsign : "MILE-AIR"
        - country : "United States"
        - iata : "Q5"
        - icao : "MLA"
        - id : 10
        - name : "40-Mile Air New"
        - type : "airline"
      - status : "success"
    - metrics
      - elapsedTime : "8.392961ms"
      - executionTime : "8.275447ms"
      - resultCount : 1
      - resultSize : 142
    - profile
      - phaseTimes
        - authorize : "1.009723ms"
        - fetch : "359.516ms"
        - instantiate : "124.756ms"
        - parse : "363.34ms"
        - plan : "179.968ms"
        - primaryScan : "6.114677ms"
        - run : "7.597287ms"
      - phaseCounts
        - fetch : 1
        - primaryScan : 1
```



```
{  
    "requestID": "c5557793-1274-4244-8d37-c3e71a03aa31",  
    "signature": {  
        "*": "*"  
    },  
    "results": [  
        {  
            "travel-sample": {  
                "callsign": "MILE-AIR",  
                "country": "United States",  
                "iata": "Q5",  
                "icao": "MLA",  
                "id": 10,  
                "name": "40-Mile Air",  
                "type": "airline"  
            }  
        }  
    ],  
    "status": "success",  
    "metrics": {  
        "elapsedTime": "21.0546ms",  
        "executionTime": "21.0546ms",  
        "resultCount": 1,  
        "resultSize": 300  
    },  
}
```

```
"profile": {
    "phaseTimes": {
        "authorize": "5.0124ms",
        "fetch": "1.0028ms",
        "parse": "1.0024ms",
        "primaryScan": "14.037ms",
        "run": "20.0522ms"
    },
    "phaseCounts": {
        "fetch": 1,
        "primaryScan": 1
    },
    "phaseOperators": {
        "authorize": 1,
        "fetch": 1,
        "primaryScan": 1
    },
    "executionTimings": {
        "#operator": "Sequence",
        "#stats": {
            "#phaseSwitches": 2,
            "kernTime": "20.0522ms"
        },
        "~children": [
        ]
    }
}
```

```
"#operator": "Authorize",
"#stats": {
    "#phaseSwitches": 4,
    "kernTime": "15.0398ms",
    "servTime": "5.0124ms"
},
"privileges": {
    "List": [
        {
            "Target": "default:travel-sample",
            "Priv": 7
        }
    ]
},
"~child": {
    "#operator": "Sequence",
    "#stats": {
        "#phaseSwitches": 3,
        "kernTime": "15.0398ms"
    },
    "~~children": [
        {
            "#operator": "Sequence",
            "#stats": {
                "#phaseSwitches": 2,
```

```
        "kernTime": "15.0398ms"
    },
    "~children": [
        {
            "#operator": "PrimaryScan",
            "#stats": {
                "#itemsOut": 1,
                "#phaseSwitches": 7,
                "execTime": "14.037ms"
            },
            "index": "def_primary",
            "keyspace": "travel-
sample",
            "limit": "1",
            "namespace": "default",
            "using": "gsi"
        },
        {
            "#operator": "Fetch",
            "#stats": {
                "#itemsIn": 1,
                "#itemsOut": 1,
                "#phaseSwitches": 11,
                "kernTime": "14.037ms",
                "servTime": "1.0028ms"
            }
        }
    ]
}
```

```
        },
        "keyspace": "travel-
sample",
        "namespace": "default"
    },
{
    "#operator": "Sequence",
    "#stats": {
        "#phaseSwitches": 5,
        "kernTime": "15.0398ms"
    },
    "~children": [
        {
            "#operator": "#stats": {
                "#itemsIn": 1,
                "#itemsOut": 1,
                "kernTime": "15.0398ms"
            },
            "result_terms": [
                {
                    "#operator": "#phaseSwitches": 9,
                    "#stats": {
                        "#itemsIn": 1,
                        "#itemsOut": 1,
                        "kernTime": "15.0398ms"
                    }
                }
            ]
        }
    ]
}
```

```
        "expr":  
    "self",  
        "star":  
true  
    }  
]  
},  
{  
    "#operator":  
"FinalProject",  
    "#stats": {  
        "#itemsIn": 1,  
        "#itemsOut": 1,  
        "#phaseSwitches": 11,  
        "kernTime":  
"15.0398ms"  
    }  
}  
]  
}  
]  
},  
{  
    "#operator": "Limit",
```

```
        "#stats": {
            "#itemsIn": 1,
            "#itemsOut": 1,
            "#phaseSwitches": 11,
            "kernTime": "15.0398ms"
        },
        "expr": "1"
    }
]
}
},
{
    "#operator": "Stream",
    "#stats": {
        "#itemsIn": 1,
        "#itemsOut": 1,
        "#phaseSwitches": 9,
        "kernTime": "20.0522ms"
    }
}
],
"~versions": [
    "2.0.0-N1QL",
    "5.0.1-5003-enterprise"
]
```

```
        }  
    }  
}
```

You can verify the `profile` section, showing phase information such as `phaseTimes`, `phaseCounts`, `phaseOperators` (respectively the time spent executing each phase, the number of documents processed per phase, and the number of operators executing each phase).

#### Profiling Using CLI:

The cbq engine must be started with authorization, for example:

```
$ ./cbq -engine=http://127.0.0.1:8091/ -u Administrator -p admin123
```

Show the statistics collected when the profile is set to phases:

```
cbq> \set -profile "phases";  
cbq> SELECT * FROM `travel-sample` WHERE type = "airline" LIMIT 1;
```

```
(base) [root@tos bin]#
(base) [root@tos bin]# cbq -engine=http://127.0.0.1:8091/ -u Administrator -p admin123
Connected to : http://127.0.0.1:8091/. Type Ctrl-D or \QUIT to exit.

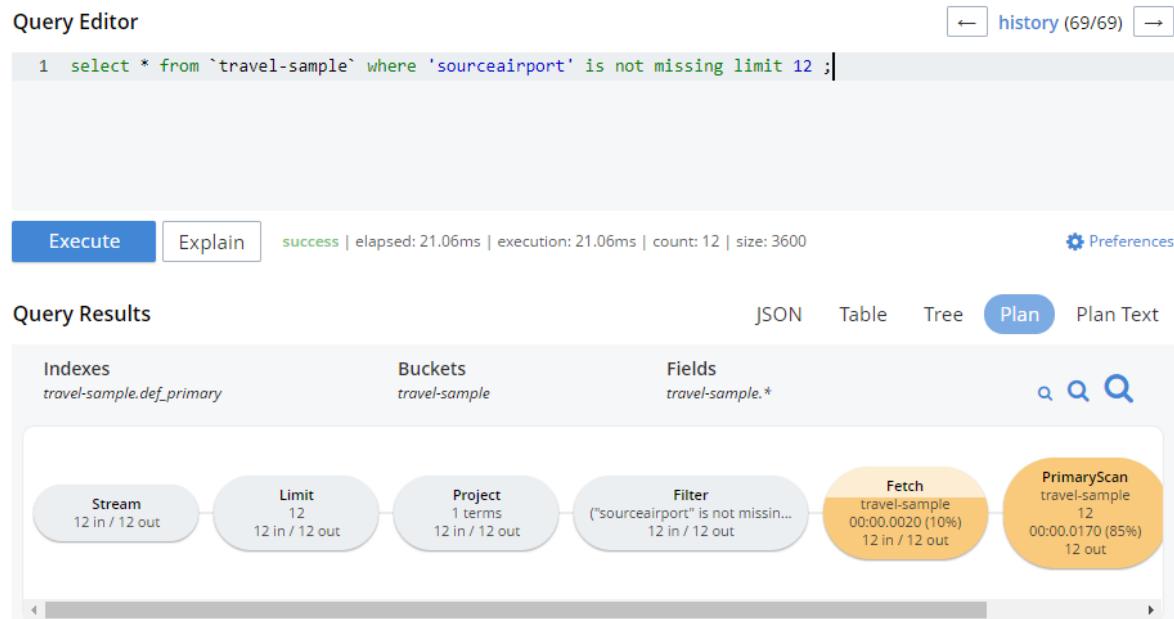
Path to history file for the shell : /root/.cbq_history
cbq> \set -profile "phases";
cbq> SELECT * FROM `travel-sample` WHERE type = "airline" LIMIT 1;
{
  "requestID": "f1c35d21-dc76-4210-bb1b-6725e100c265",
  "signature": {
    "*": "*"
  },
  "results": [
    {
      "travel-sample": {
        "callsign": "MILE-AIR",
        "country": "United States",
        "iata": "Q5",
        "icao": "MLA",
        "id": 10,
        "name": "40-Mile Air New",
        "type": "airline"
      }
    }
  ],
  "status": "success",
  "metrics": {
    "elapsedTime": "6.039602ms",
    "latency": {
      "min": "0.000000ms",
      "max": "6.039602ms",
      "avg": "6.039602ms"
    }
  }
}
```

## Visual profiles

Access detailed profiling information for individual statements, without having to do anything special, is through the Query tab in the Admin Console UI.

Query WorkBench turns profiling on by default, and upon completion of the request, timings and document counts are readily available in pictorial form, by clicking on the “plan” button.

```
select * from `travel-sample` where 'sourceairport' is not missing limit 12 ;
```



Query WorkBench turns profiling on by default, and upon completion of the request, timings and document counts are readily available in pictorial form, by clicking on the “plan” button. The operators are neatly colour coded by cost, and hovering over individual operators will result in copious amounts of raw timings popping up.

The Preferences allows to change profiling settings, as well as other settings such as named and positional parameters.

Hover mouse on Any of the Plan and you can verify the details.

A screenshot of the Couchbase Query Profiler interface. On the left, there's a tree view with nodes like 'Fields' and 'PrimaryScan3'. A yellow tooltip is overlaid on the 'PrimaryScan3' node, containing the following information:

**PrimaryScan3**  
12  
00:00.0023 (26.2%)  
12 out

The main pane shows a detailed breakdown of the 'PrimaryScan3' plan:

```
PrimaryScan3
#stats
  o #itemsOut - 12
  o #phaseSwitches - 51
  o execTime - 51.627µs
  o kernTime - 5.367µs
  o servTime - 2.279549ms
index - #primary
index_projection
  o primary_key - true
keyspace - travel-sample
limit - 12
namespace - default
using - gsi
#time_normal - 00:00.0023
#time_absolute - 0.0023311759999999995
```

The 'Plan' tab is selected at the top right of the main pane.

## Verify the primaryscan

We can derived the following out of the above metrics:

- primary scan consumed **26.2%** of the total query execution.

- the number of documents digested (#itemsOut),
- time spent executing operator code (execTime),
- waiting to be scheduled (kernTime)
- and waiting for services to supply data (servTime),
- the number of times the operator has changed state (executing, waiting to be scheduled, waiting for data - PhaseSwitches),

Scenario, if stuck in “kernel”, then maybe a consumer down the pipeline is not accepting data, or the machine might be so loaded that the scheduler is not actually finding time slices for the operator.  
Any operator: high kernel time

This is a bottleneck in an operator further down the pipeline.

Look for an operator with high execution or services time.

If one can't be found, the runtime kernel is struggling to schedule operators: the query node is overloaded and adding a new one might come handy.

### **Fetches: high number of documents ingested**

A good query plan should use an index that produces only the document keys that are relevant for the request.

If the number of keys passed to the Fetch operator is much more than the final number of documents returned to the client, the index chosen is not selective enough.

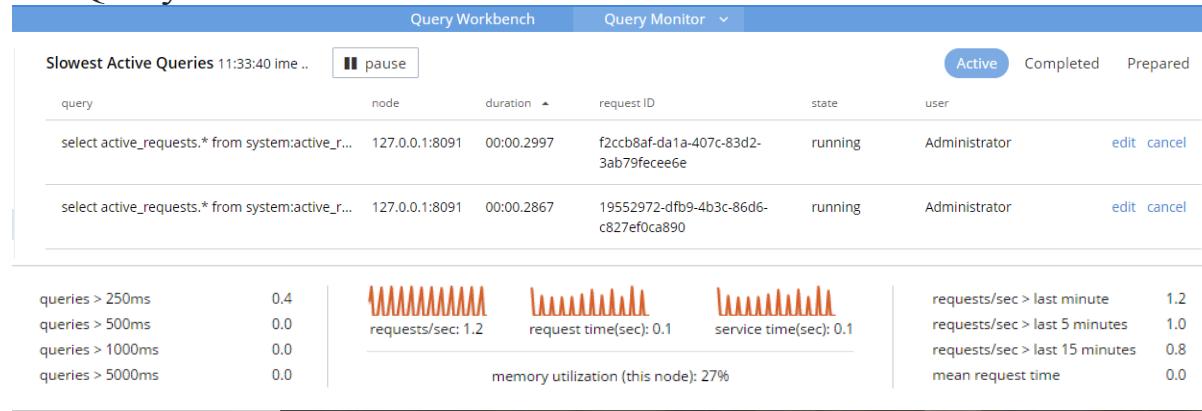
### **Scans: high number of document produced**

This is the counterpart of the previous symptom: the index chosen is not selective enough. Consider creating an index so additional predicates can be pushed to the index scan.

## Query Monitoring

Starting from version 5.0, the Query tab of the UI sports a Query Monitoring view providing live information on currently running requests, completed slow running requests and prepared statements:

The view refreshes automatically, and offers the option to cancel requests and copy request texts into the Query Editor.



You can also monitor the above from REST API as shown below:

```
curl http://localhost:8093/admin/vitals -u Administrator -p admin123
curl http://localhost:8093/admin/active_requests -u Administrator
curl http://localhost:8093/admin/completed_requests -u Administrator
curl http://localhost:8093/admin/prepareds -u Administrator
```

```
C:\Users\henryp>curl http://localhost:8093/admin/vitals -u Administrator
Enter host password for user 'Administrator':
{"uptime":"26h12m44.371873s","local.time":"2018-03-21 11:51:36.047595 +0530 IST","version":"2.0.0-N1QL","total.threads":155,"cores":4,"gc.num":22874448,"gc.pause.time":"860.6236ms","gc.pause.percent":0.0001,"memory.usage":19364032,"memory.total":6764258560,"memory.system":68282616,"cpu.user.percent":0,"cpu.sys.percent":0,"request.completed.count":2711,"request.active.count":0,"request.per.sec.1min":0.8,"request.per.sec.5min":0.8054,"request.per.sec.15min":0.8015,"request_time.mean":"43.50872ms","request_time.median":"15.0379ms","request_time.80percentile":"21.05588ms","request_time.95percentile":"289.872024ms","request_time.99percentile":"435.578131ms","request.prepared.percent":0}
C:\Users\henryp>curl http://localhost:8093/admin/active_requests -u Administrator
Enter host password for user 'Administrator':
[]
C:\Users\henryp>curl http://localhost:8093/admin/completed_requests -u Administrator
Enter host password for user 'Administrator':
[{"elapsedTime":6.7479421s,"errorCount":0,"phaseCounts":{"fetch":31591,"primaryScan":31591},"phaseOperators":{"authorize":1,"fetch":1,"primaryScan":1},"requestId":"be9079f8-c9f9-4007-907e-083b7209a942","requestTime":"2018-03-21 10:03:28.798225 +0530 IST","resultCode":31591,"resultSize":36764944,"scanConsistency":"unbounded","serviceTime":6.7479421s,"state":"completed","statement":"select * from `travel-sample`","users":"Administrator"}, {"elapsedTime":2.4936289s,"errorCount":0,"phaseOperators":{"authorize":1,"count":1},"requestId":"b93e176b-68ef-4d30-8f69-f203a03971e5","requestTime":"2018-03-21 11:44:04.2959821 +0530 IST","resultCode":1,"resultSize":32,"scanConsistency":"unbounded","serviceTime":2.4936289s,"state":"completed","statement":"select count(*) cnt from `default`","users":"Administrator"}, {"elapsedTime":3.4652148s,"errorCount":0,"phaseOperators":{"authorize":1,"count":1},"requestId":"42d2815d-7c96-48dd-9536-29c103e0c27a","requestTime":"2018-03-21 11:43:50.2918691 +0530 IST","resultCode":1,"resultSize":37,"scanConsistency":"unbounded","serviceTime":3.4652148s,"state":"completed","statement":"select count(*) cnt from `couchmusic2`","users":"Administrator"}, {"elapsedTime":10.0737057s,"errorCount":0,"phaseCounts":{"fetch":31591,"primaryScan":31591}, "phaseOperators":{"authorize":1,"fetch":1,"primaryScan":1}, "requestId": "3ec6292d-9eb4-44db-9900-0d4cd5d0dd62", "requestTime": "2018-03-21 11:06:30.41534 +0530 IST", "resultCode": 31591, "resultSize": 107818442, "scanConsistency": "unbounded", "serviceTime": 10.0737057s} ]
```

## Preparation

We will use the beer-sample bucket in order to provide some load for the following examples.

Create a secondary index on `beer-sample`(`city)

```
CREATE INDEX Index_tos_beer_city ON `beer-sample`(`city);
```

### Query Editor

```
1 CREATE INDEX Index_tos_beer_city ON `beer-sample`(city);
```

Execute

Explain

✓ success | elapsed: 4.82s | execution: 4.82s |

*Start up a few client applications preparing*

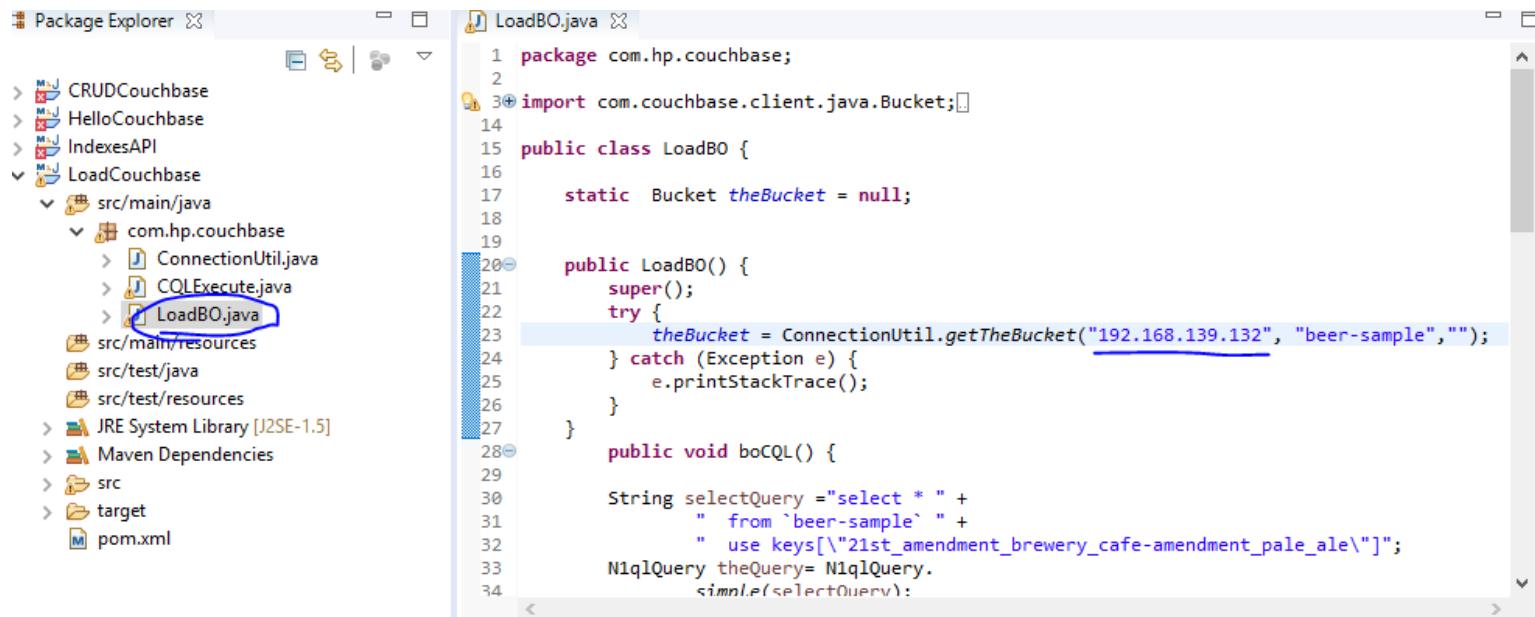
```
select *
  from `beer-sample`
 use keys["21st_amendment_brewery_cafe-amendment_pale_ale"];
```

*and*

```
select *
  from `beer-sample`
 where city = "Lawton";
```

And then executing the statements above for, say, 10000 times per client.

Import the LoadCouchbase java project in your eclipse workspace and update the IP of the couchbase before executing it.



```
1 package com.hp.couchbase;
2
3 import com.couchbase.client.java.Bucket;
4
5 public class LoadBO {
6
7     static Bucket theBucket = null;
8
9
10    public LoadBO() {
11        super();
12        try {
13            theBucket = ConnectionUtil.getTheBucket("192.168.139.132", "beer-sample","");
14        } catch (Exception e) {
15            e.printStackTrace();
16        }
17    }
18
19    public void boCQL() {
20
21        String selectQuery ="select * " +
22            " from `beer-sample` " +
23            " use keys['21st_amendment_brewery_cafe-amendment_pale_ale\\']";
24        N1qlQuery theQuery= N1qlQuery.
25            simple(selectQuery);
26
27    }
28
29
30
31
32
33
34 }
```

[D:\CodeWS\LoadCouchbase]

Execute the following method it will fire a numerous Queries to your couchbase server.

```

1 package com.hp.couchbase;
2
3 import rx.Observable;
4
5 public class CQLExecute {
6
7     public static void main(String[] args) {
8         try {
9             findBeersByCQL();
10            //findbeersbycqlIII();
11        } catch (Exception e) {
12            // TODO Auto-generated catch block
13            e.printStackTrace();
14        }
15    }
16
17    static void findBeersByCQL() throws Exception{
18        final LoadBO bo = new LoadBO();
19        Thread findThread = new Thread() {
20            public void run() {
21                int i=1000000;
22                while(i >=0 ) {
23                    bo.boCQL();
24                    bo.boCQLIII();
25                    i--;
26                    System.out.println(i);
27                }
28            }
29        };
30    }
31}

```

You can verify the request from the console using the query Monitor.

Query → Query Monitor → Current Request.

Slowest Completed Queries 16:16:20 lme ...							
query	node	duration	result count	state	run at	user	
<input type="button" value="pause"/> Active <input checked="" type="button" value="Completed"/> <span style="border: 1px solid #ccc; padding: 2px;">Completed</span> Prepared							
select * from `travel-sample` where `sourceairpo...	127.0.0.1:8091	00:10.9857	31591	completed	15:24:41 IST	Administrator	<a href="#">edit</a>
select * from `travel-sample` where `sourceairpo...	127.0.0.1:8091	00:06.8592	31591	completed	15:16:45 IST	cbuser	<a href="#">edit</a>
select * from `travel-sample` where `sourceairpo...	127.0.0.1:8091	00:06.1443	31591	completed	15:21:25 IST	cbuser	<a href="#">edit</a>

Execute the following query in the Query Editor:

```
select active_requests.* from system:active_requests
```

Query Editor ← history (5/5) →

```
1 select active_requests.* from system:active_requests
```

Execute Explain ✓ success | elapsed: 325.09ms | execution: 325.00ms | count: 2 | preferences  
size: 2314

Query Results  JSON Table Tree Plan Plan Text

```
1 [  
2 {  
3   "elapsedTime": "509.165027ms",  
4   "executionTime": "509.026108ms",  
5   "node": "127.0.0.1:8091",  
6   "phaseCounts": {  
7     "fetch": 1088,  
8     "primaryScan": 1515
```

Copy the Json value in a text file and verify it content.

```
[  
{  
  "elapsedTime": "509.165027ms",  
  "executionTime": "509.026108ms",  
  "node": "127.0.0.1:8091",  
  "phaseCounts": {  
    "fetch": 1088,  
    "primaryScan": 1515
```

```
},
"phaseOperators": {
    "authorize": 1,
    "fetch": 1,
    "join": 1,
    "primaryScan": 1
},
"remoteAddr": "192.168.139.1:59457",
"requestId": "9ee046ed-664d-48af-b6a1-fd4a8a5e5a1a",
"requestTime": "2019-07-16 17:19:19.367174783 +0530 IST",
"scanConsistency": "unbounded",
"state": "running",
"statement": "select b.type,b.category ,br.name FROM `beer-sample` as b join `beer-sample` as br on keys b.brewery_id where b.type=='beer'",
"userAgent": "couchbase-java-client/2.5.5 (git: 2.5.5, core: 1.5.5) (Windows 10/10.0 amd64; Java HotSpot(TM) 64-Bit Server VM 1.8.0_171-b11)",
"users": "cbuser,cbuser"
},
{
"clientContextID": "f46f93c2-4f5e-401a-857a-e7fb3b73bf7f",
"elapsedTime": "323.938551ms",
"executionTime": "323.844971ms",
"node": "127.0.0.1:8091",
"phaseCounts": {
    "primaryScan": 2
```

```
},
"phaseOperators": {
    "authorize": 1,
    "fetch": 1,
    "primaryScan": 1
},
"phaseTimes": {
    "authorize": "13.11335ms",
    "fetch": "34.609μs",
    "instantiate": "59.82μs",
    "parse": "603.153μs",
    "plan": "132.396μs",
    "primaryScan": "292.127665ms"
},
"remoteAddr": "127.0.0.1:52092",
"requestId": "f1096838-9930-4736-8965-fdoc4e4c005b",
"requestTime": "2019-07-16 17:19:19.552428929 +0530 IST",
"scanConsistency": "unbounded",
"state": "running",
"statement": "select active_requests.* from system:active_requests",
"userAgent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/74.0.3729.169 Safari/537.36 (Couchbase Query Workbench (6.0.1-2037-enterprise))",
"users": "Administrator"
}
```

]

You can verify what all queries are running in the moment.

## Too much load on the node

Among the useful information reported by the system:active\_requests keyspace there are two pieces of timing information:

- Elapsed time
  - Execution time

```
{  
  "elapsedTime": "509.165027ms",  
  "executionTime": "509.026108ms",  
  "node": "127.0.0.1:8091",  
  "phaseCounts": {  
    "fetch": 1088,  
    "primaryScan": 1515  
  },  
  "primary": true,  
  "secondary": false  
}
```

The first starts being accrued as soon as the N1QL service becomes aware of the request, the second as soon as the request is being executed.

Under normal circumstances there should be very little difference in between the two times, so the following query...

```
select avg(100*
          (str_to_duration(elapsedTime) - str_to_duration(executionTime)) /
          str_to_duration(elapsedTime)), count(*)
from system:active_requests;
```

...should return only a few percentage points (say 1 to 3%):

```
{
  "requestID": "137b5777-d420-45e9-bbef-0b15197c9a93",
  "signature": {
    "$1": "number",
    "$2": "number"
  },
  "results": [
    {
      "$1": 3.210601953400783,
      "$2": 8
    }
  ],
  "status": "success",
  "metrics": {
    "elapsedTime": "1.938543ms",
    "executionTime": "1.910804ms",
    "resultCount": 1,
    "resultSize": 68
  }
}
```

```
}
```

The query itself should execute quite quickly. When the server is **under stress**, requests will most probably be queued before being executed **and the ratio of elapsed to execution time will increase** — to put it in perspective a value of 50 means that the request is spending just as much time queued as being executed.

In general terms, once the node is under stress, the query above will plateau at a specific percentage point.

On my development machine this is about 25% (although higher spikes will make an appearance from time to time), however, a telltale sign of the node being under stress is that the elapsed to execution ratio of the query itself is quite high:

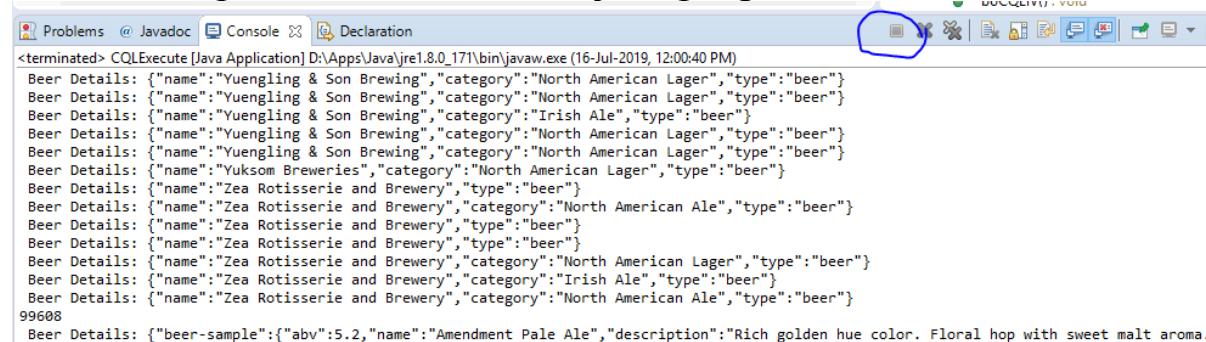
```
{
  "requestID": "8cafcd14-fbdb-48bd-a35d-9f0fb03752ae",
  "signature": {
    "$1": "number",
    "$2": "number"
  },
  "results": [
    {
      "$1": 35.770020645707156,
      "$2": 94
    }
  ],
}
```

```

    "status": "success",
    "metrics": {
        "elapsedTime": "29.999017ms",
        "executionTime": "3.864563ms",
        "resultCount": 1,
        "resultSize": 48
    }
}

```

You can stop the execution of the java program.



```

Problems Javadoc Console Declaration
terminated> CQLExecute [Java Application] D:\Apps\Java\jre1.8.0_171\bin\javaw.exe (16-Jul-2019, 12:00:40 PM)
Beer Details: {"name":"Yuengling & Son Brewing","category":"North American Lager","type":"beer"}
Beer Details: {"name":"Yuengling & Son Brewing","category":"North American Lager","type":"beer"}
Beer Details: {"name":"Yuengling & Son Brewing","category":"Irish Ale","type":"beer"}
Beer Details: {"name":"Yuengling & Son Brewing","category":"North American Lager","type":"beer"}
Beer Details: {"name":"Yuengling & Son Brewing","category":"North American Lager","type":"beer"}
Beer Details: {"name":"Yuksom Breweries","category":"North American Lager","type":"beer"}
Beer Details: {"name":"Zea Rotisserie and Brewery","type":"beer"}
Beer Details: {"name":"Zea Rotisserie and Brewery","category":"North American Ale","type":"beer"}
Beer Details: {"name":"Zea Rotisserie and Brewery","type":"beer"}
Beer Details: {"name":"Zea Rotisserie and Brewery","type":"beer"}
Beer Details: {"name":"Zea Rotisserie and Brewery","category":"North American Lager","type":"beer"}
Beer Details: {"name":"Zea Rotisserie and Brewery","category":"Irish Ale","type":"beer"}
Beer Details: {"name":"Zea Rotisserie and Brewery","category":"North American Ale","type":"beer"}
99608
Beer Details: {"beer-sample":{"abv":5.2,"name":"Amendment Pale Ale","description":Rich golden hue color. Floral hop with sweet malt aroma.

```

Click on the above Icon.

Another important piece of information reported by both system:active\_requests and system:completed\_requests is the number of documents processed by each execution phase.

This information is interesting for at least two different reasons.

The first is, even though the system keyspaces do not report a request plan, and therefore individual query paths are not known, we can still draw a fair amount of information from the fact that specific phases are used.

For example, primary scans are the N1QL equivalent of RDBMS's sequential scans - unless very small buckets are involved, where an ad hoc index adds no performance benefit, primary scans are best avoided.

Similarly, you may have already heard that *covering indexes* (indexes whose keys encompass all the expressions in the select list) are preferable to non covering indexes, because when the index chosen is not covering, request execution will require a Fetch phase after the index scan, in order to access each document, which clearly requires further effort on the N1QL service part.

The mere presence of a Fetch phase is a dead giveaway that covering indexes are not being used.

The second reason, more obvious, is that document counts instantly give a sense of the cost of individual phases, and by extension, individual requests.

We will now explore a few use cases exploiting phase counts.

#### Preparation

From cbq or using Query Editor, execute a few queries. Change directory to the bin folder of couchbase installation and execute the cbq.

```
# cbq -u=Administrator -p admin123
```

```
(base) [root@tos bin]#
(base) [root@tos bin]# pwd
/opt/couchbase/bin
(base) [root@tos bin]# cbq -u=Administrator -p admin123
Connected to : http://localhost:8091/. Type Ctrl-D or \QUIT to exit.

Path to history file for the shell : /root/.cbq_history
cbq> 
```

```
select * from `travel-sample` limit 1000;
```

```
        "flight": "AF099",
        "utc": "12:55:00"
    },
    {
        "day": 6,
        "flight": "AF796",
        "utc": "11:54:00"
    }
],
"sourceairport": "TLV",
"stops": 0,
"type": "route"
}
}
],
"status": "success",
"metrics": {
    "elapsedTime": "27.415153932s",
    "executionTime": "27.415019296s",
    "resultCount": 31591,
    "resultSize": 107818442
}
}
cbq> select * from `travel-sample`;
```

The screenshot shows the Couchbase Query Editor interface. At the top, there's a header with "Query Editor" on the left and "history (129/129)" on the right. Below the header is a code editor containing the query: "1 select \* from `travel-sample` limit 1000;". Underneath the code editor, there are two buttons: "Execute" (which is highlighted in blue) and "Explain". To the right of these buttons, the status is shown as "✓ success | elapsed: 195.33ms | execution: 195.21ms | count: 1000 | size: 464543". Further to the right is a "preferences" link. Below the status, there are four tabs: "JSON" (which is selected and highlighted in blue), "Table", "Tree", and "Plan Text". The main area below the tabs displays the query results in JSON format, showing a single document from the "travel-sample" database.

```

1 [ { "travel-sample": { "callsign": "MILE-AIR", "country": "United States", "iata": "Q5", "icao": "MLA", "id": 10, "name": "40-Mile Air New" } }

```

## Requests Using Primary Scans

As stated earlier, all we are looking for is requests where the PhaseCounts subdocuments has a PrimaryScan field – Execute in the Query Editor:

```
select *
  from system:completed_requests
  where phaseCounts.`primaryScan` is not missing;
```

Which in our case yields:

```
[ { "completed_requests": { }
```

```
"elapsedTime": "1.8890459s",
"errorCount": 0,
"node": "127.0.0.1:8091",
"phaseCounts": {
    "fetch": 7306,
    "primaryScan": 7306
},
"phaseOperators": {
    "authorize": 1,
    "fetch": 1,
    "join": 1,
    "primaryScan": 1
},
"remoteAddr": "127.0.0.1:65391",
"requestId": "484f3e02-1f36-4739-93b5-ff012da4654d",
"requestTime": "2018-03-21 16:45:58.837828 +0530 IST",
"resultCount": 5892,
"resultSize": 411161,
"scanConsistency": "unbounded",
"serviceTime": "1.8890459s",
"state": "completed",
"statement": "select b.type,b.category ,br.name  FROM `beer-sample`  as b join `beer-sample`  as br  on keys b.brewery_id      where b.type=='beer'",
```

```
        "userAgent": "couchbase-java-client/2.5.5 (git: 2.5.5, core: 1.5.5) (Windows 10/10.0 amd64; Java HotSpot(TM) 64-Bit Server VM 9.0.4+11)",
        "users": "cbuser"
    }
},
{
    "completed_requests": {
        "elapsedTime": "1.2402974s",
        "errorCount": 0,
        "node": "127.0.0.1:8091",
        "phaseCounts": {
            "fetch": 7306,
            "primaryScan": 7306
        },
        "phaseOperators": {
            "authorize": 1,
            "fetch": 1,
            "join": 1,
            "primaryScan": 1
        },
        "remoteAddr": "127.0.0.1:65391",
        "requestId": "446447bf-5fe2-46cb-abf3-041eb9ae8b51",
        "requestTime": "2018-03-21 16:46:21.0232429 +0530 IST",
        "resultCount": 5892,
```

```
"resultSize": 411161,
"scanConsistency": "unbounded",
"serviceTime": "1.2402974s",
"state": "completed",
"statement": "select b.type,b.category ,br.name  FROM `beer-sample`  as b join `beer-sample`  as br  on keys b.brewery_id  where b.type=='beer'",
"userAgent": "couchbase-java-client/2.5.5 (git: 2.5.5, core: 1.5.5) (Windows 10/10.0 amd64; Java HotSpot(TM) 64-Bit Server VM 9.0.4+11)",
"users": "cbuser"
},
{
"completed_requests": {
"elapsedTime": "1.3625203s",
"errorCount": 0,
"node": "127.0.0.1:8091",
"phaseCounts": {
"fetch": 7306,
"primaryScan": 7306
},
"phaseOperators": {
"authorize": 1,
"fetch": 1,
```

```
        "join": 1,
        "primaryScan": 1
    },
    "remoteAddr": "127.0.0.1:65391",
    "requestId": "b7b9867b-0240-419b-b0f6-7e83a0995b7f",
    "requestTime": "2018-03-21 16:47:11.8169387 +0530 IST",
    "resultCount": 5892,
    "resultSize": 411161,
    "scanConsistency": "unbounded",
    "serviceTime": "1.3625203s",
    "state": "completed",
    "statement": "select b.type,b.category ,br.name  FROM `beer-sample`  as b join `beer-sample`  as br  on keys b.brewery_id  where b.type=='beer'",
    "userAgent": "couchbase-java-client/2.5.5 (git: 2.5.5, core: 1.5.5) (Windows 10/10.0 amd64; Java HotSpot(TM) 64-Bit Server VM 9.0.4+11)",
    "users": "cbuser"
}
],
]
```

## Preparation

The example below requires a secondary index on `travel-sample`(type)

Execute the following query:

```
select id
  from `travel-sample`
 where type="route" and any s in schedule satisfies s.flight="AZ917" end;
```

The screenshot shows the Couchbase Query Editor interface. At the top, there is a "Query Editor" header with a "history (14/14)" button. Below it is a code editor containing the following query:

```
1 select id
2   from `travel-sample`
3 where type="route" and any s in schedule satisfies s.flight="AZ917" end;
```

Below the code editor are two buttons: "Execute" (highlighted in blue) and "Explain". To the right of these buttons is a status message: "✓ success | elapsed: 3.42s | execution: 3.42s | count: 10 | size: 350" followed by a "preferences" link.

At the bottom, there is a "Query Results" section with a "JSON" button (highlighted in blue), followed by "Table", "Tree", "Plan", and "Plan Text" buttons. The JSON results pane displays the following output:

```
1 [
2   {
3     "id": 12913
4   },
5 ]
```

Verify the Index that is use in the above query execution. i.e Primary

The screenshot shows the Couchbase Query Planner interface. A query is being executed:

```

2   from travel-sample
3   where type="route" and
      IndexScan3
      index - def_type
      index_id - cfa92e978af70468
      index_projection
        o primary_key - true
      keyspace - travel-sample
      namespace - default
      spans
        o exact - true
        o range
          o high - "route"
          o inclusion - 3
          o low - "route"
      using - gsi
      es s.flight="AZ917" end;

```

The 'Explain' button is highlighted. The results pane shows the execution time and count:

execution: 2.06ms | count: 1

Below the results, there are tabs for Table, Tree, Plan, and Plan Text. The 'Plan' tab is selected.

## Requests not Using Covering Indexes

In a similar way, we can find non covering indexes by stipulating that the requests we are looking for have **both index scans** and **fetches** in the PhaseCounts subdocument:

```

select *
  from system:completed_requests
 where phaseCounts.`indexScan` is not missing
   and phaseCounts.`fetch` is not missing;

```

Which, in our case yields:

```
[
{
  "completed_requests": {

```

```
        "clientContextID": "e5f0e69a-ff3e-47e4-90c7-881772c2e1b7",
        "elapsedTime": "2.877312961s",
        "errorCount": 0,
        "phaseCounts": {
            "fetch": 24024,
            "indexScan": 24024
        },
        "phaseOperators": {
            "fetch": 1,
            "indexScan": 1
        },
        "requestId": "12402dd8-430a-4c32-84c1-a59b88695e66",
        "resultCount": 10,
        "resultSize": 350,
        "serviceTime": "2.8772514s",
        "state": "completed",
        "statement": "select id from `travel-sample` where type=\"route\" and any s in schedule satisfies s.flight=\"AZ917\" end;",
        "time": "2016-11-10 13:10:00.04988112 -0800 PST"
    }
}
]
```

Create a secondary index on the type.

```
CREATE INDEX Index_tos_travel ON `travel-sample`(type);
```

Now execute the following query again.

Query Editor

← history (14/16) →

```
1 select id
2   from `travel-sample`
3 where type="route" and any s in schedule satisfies s.flight="AZ917" end;
```

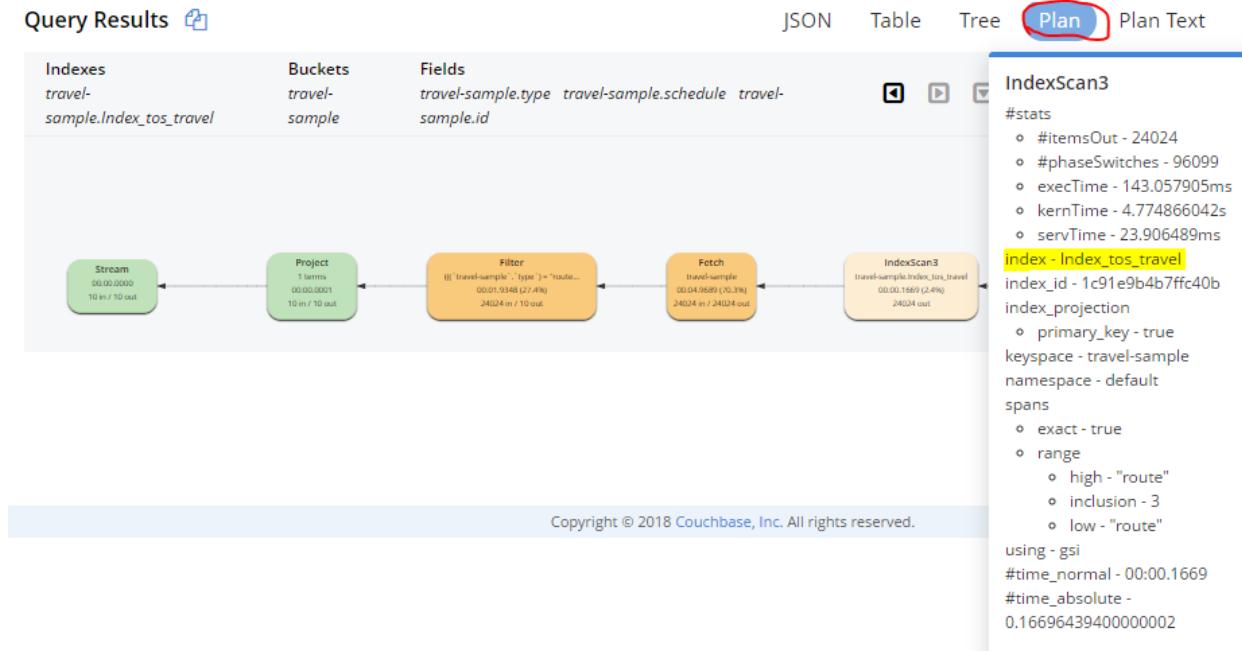
Execute Explain ✓ success | elapsed: 5.08s | execution: 5.08s | count: 10 | size: 350 ⚙ preferences

Query Results 

JSON Table Tree Plan Plan Text

```
1 [
2 {
3   "id": 12913
4 },
5 ]
```

Verify the plan:



This shows that secondary index is being used for this query execution.

### Requests Using a Poorly Selective Index

Loosely speaking, Selectivity is that property of a predicate which indicates how good that predicate is at isolating those documents which are relevant for the query result set.

In general terms, when using an index scan, we would like to have the scan return only the relevant document keys, and no more, so that later stages of the plan (eg, fetch or filter) only have to operate on relevant documents.

A simple measure of poor selectivity of a predicate is the total number of keys (or documents) to be considered by the predicate divided by the number of documents that

satisfy the predicate - the higher the number, the worse the selectivity (in general terms selectivity is defined as the number qualifying items divided by the total number of items).

Clearly, when an index path is in use, we would like that the relevant filters are as relevant as possible, and we can determine if this is the case by just measuring the number of documents fetched against the number of documents returned.

The query below finds requests that do use indexes and that have a number of fetches to results ratio greater than ten.

This means that the request fetches ten documents for each document that satisfies all the predicates in the where clause: you threshold may very well be different (and lower, I would suggest!).

```
select phaseCounts.`fetch` / resultCount efficiency, *
  from system:completed_requests
  where phaseCounts.`indexScan` is not missing
    and phaseCounts.`fetch` / resultCount > 10
  order by efficiency desc;
```

In our case, this yields:

```
[{"completed_requests": {"clientContextID": "e5f0e69a-ff3e-47e4-90c7-881772c2e1b7", "elapsedTime": "2.877312961s", "errorCount": 0, "fetch": 10, "indexScan": true, "key": "1", "resultCount": 1, "requestID": "1", "startKey": "1", "totalBytes": 1000}}]
```

```
"phaseCounts": {
    "fetch": 24024,
    "indexScan": 24024
},
"phaseOperators": {
    "fetch": 1,
    "indexScan": 1
},
"requestId": "12402dd8-430a-4c32-84c1-a59b88695e66",
"resultCount": 10,
"resultSize": 350,
"serviceTime": "2.8772514s",
"state": "completed",
"statement": "select id from `travel-sample` where type=\"route\" and any s in schedule satisfies s.flight=\"AZ917\" end;",
"time": "2016-11-10 13:10:00.04988112 -0800 PST"
},
"efficiency": 2402.4
}
]
```

## Determining Requests With the Highest Cost

As we were saying earlier, PhaseCounts give a ready sense of individual phases cost. Sum them up, and you can get a measure of the cost of individual requests.

In the statement below we turn the PhaseCounts subdocument into an array in order to tot up the cost:

```
select statement, array_sum(object_values(phaseCounts)) cost
from system:completed_requests
order by cost desc;
```

This returns the statements we run earlier in order of decreasing cost:

```
[  
  {  
    "statement": "select * from `travel-sample`",  
    "cost": 63182  
  },  
  {  
    "statement": "select * from `travel-sample`",  
    "cost": 63182  
  },  
  {  
    "statement": "select * from `travel-sample`",  
    "cost": 63182  
  },  
  {  
    "statement": "select id from `travel-sample` where type  
= \"route\" and any s in schedule satisfies s.flight= \"AZ917\" end",  
    "cost": 48048  
  }]
```

]

## Aggregating Execution Times for Completed Requests

Determining total execution times for individual statement texts is not only the preserve of prepared statements.

A very similar technique can be used on system:completed\_requests.

Further, this can be combined with filtering for suboptimal paths.

The following query looks for requests having used a primary scan, returning several aggregates for each individual statement:

```
select count(*) num,
       min(serviceTime) minEx, max(serviceTime) maxEx,
       avg(str_to_duration(serviceTime)/1000000000) avgEx,
       statement
  from system:completed_requests
 where phaseCounts.primaryScan is not missing
 group by statement;
```

Note that execution times are strings, and in principle they should be turned into durations in order to be aggregated properly. This is what we are doing for avg():

[  
{

```
"statement": "select * from `travel-sample`",
```

```

    "avgEx": 38.19585500033333,
    "maxEx": "42.17955482s",
    "minEx": "35.638946836s",
    "num": 3
}
]

```

## Canceling Requests

We have seen before that a rogue request can be cancelled using the REST API.

This can be achieved through N1QL directly by simply deleting from system:active\_requests:

```
delete from system:active_requests
where requestId="d3b607d4-1ff1-49c8-994e-8b7e6828f6a0";
```

(replace the request id in quotes with the RequestId of the request you wish to cancel).

But while REST can operate only on a request by request basis, N1QL allows for more creative operations. For example, we can stop all executing instances of a specific statement:

```
delete from system:active_requests
where statement = "select * from `travel-sample`";
```

Similarly, we can stop all instances of a specific prepared statement:

```
delete from system:active_requests
```

```
where `prepared.Name` = "s1";
```

In all of the examples above, please be mindful that we are using an equality predicate on the statement text: if you have several variations of the same statement (eg lower case vs upper case, or different amounts of spacing, or...), you'll either have to match them individually, or have a more comprehensive filter.

-----Lab Ends Here -----

## 28. Advance command:

### Uninstalling on RHEL

```
service couchbase-server stop or systemctl stop couchbase-server
rpm -e couchbase-server
rm -fr /opt/*
```

### Views Code:

```
brewery
b_city
function (doc, meta) {
if(doc.type == 'brewery' && doc.city)
  emit(doc.city, null);
}
```

### Specific\_Attribute

```
function (doc, meta) {  
    if(doc.type == "beer" && doc.brewery_id) {  
        emit(doc.brewery_id,doc.abv)  
    }  
}  
  
function (doc, meta) {  
    if(doc.type == "brewery" && doc.city) {  
        emit(doc.city,null)  
    }  
}  
  
function (doc, meta) {  
    if(doc.country,doc.state, doc.city) {  
        emit([doc.country,doc.state, doc.city],1);  
    } else if(doc.country,doc.state) {  
        emit([doc.country,doc.state],1);  
    } else if(doc.country) {  
        emit([doc.country],1);  
    }  
}
```

## Mount Shared Folder

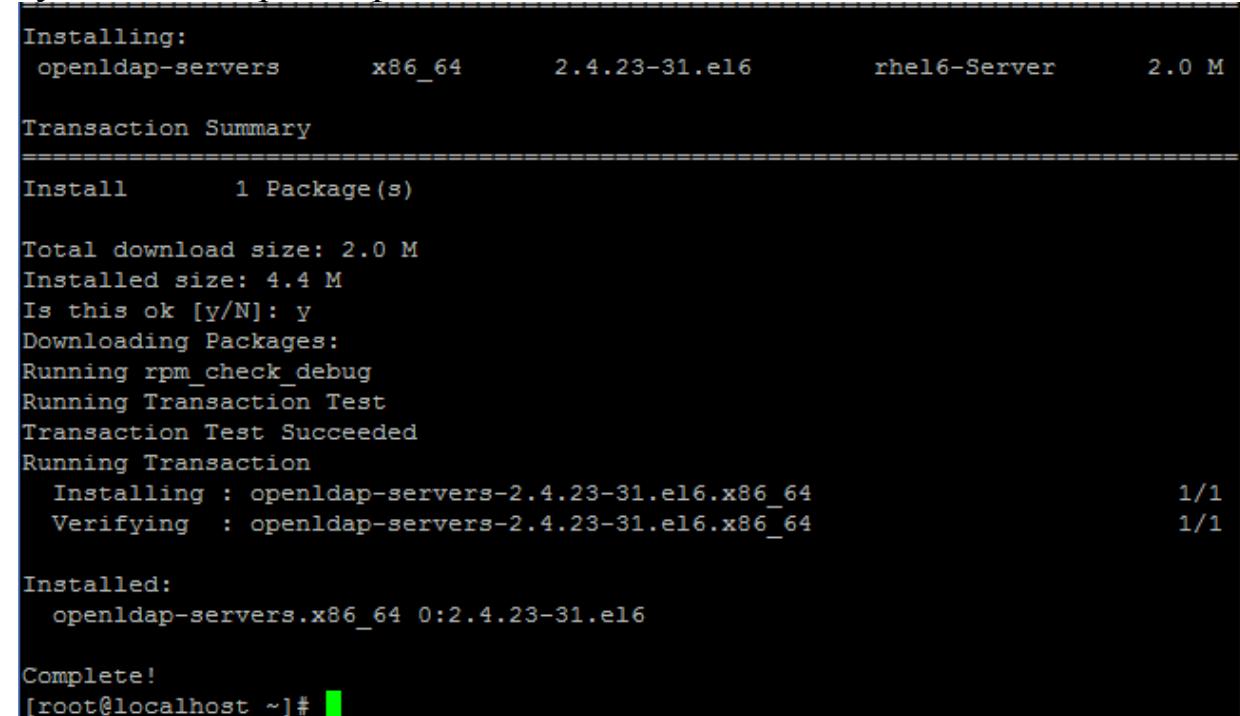
```
/usr/bin/vmhgfs-fuse .host:/ /mnt/hgfs -o subtype=vmhgfs-fuse,allow_other
```

### 1. OpenLdap Installation

#### Linux

Note: Ensure to enable YUM before proceeding ahead.

```
#yum install openldap-servers
```



```
Installing:
  openldap-servers      x86_64      2.4.23-31.el6      rhel6-Server      2.0 M

Transaction Summary
=====
Install      1 Package(s)

Total download size: 2.0 M
Installed size: 4.4 M
Is this ok [y/N]: y
Downloading Packages:
Running rpm_check_debug
Running Transaction Test
Transaction Test Succeeded
Running Transaction
  Installing : openldap-servers-2.4.23-31.el6.x86_64          1/1
  Verifying  : openldap-servers-2.4.23-31.el6.x86_64          1/1

Installed:
  openldap-servers.x86_64 0:2.4.23-31.el6

Complete!
[root@localhost ~]#
```

```
#yum install openldap-clients
```

```

=====
Installing:
  openldap-clients      x86_64      2.4.23-31.el6      rhel6-Server      165 k

Transaction Summary
=====
Install      1 Package(s)

Total download size: 165 k
Installed size: 609 k
Is this ok [y/N]: y
Downloading Packages:
Running rpm_check_debug
Running Transaction Test
Transaction Test Succeeded
Running Transaction
  Installing : openldap-clients-2.4.23-31.el6.x86_64          1/1
  Verifying  : openldap-clients-2.4.23-31.el6.x86_64          1/1

Installed:
  openldap-clients.x86_64 0:2.4.23-31.el6

Complete!
[root@localhost ~]# 

```

### Create olcRootDN Account as Admin

It is always recommended to create a dedicated user account first with the full permissions to change information on the LDAP database.

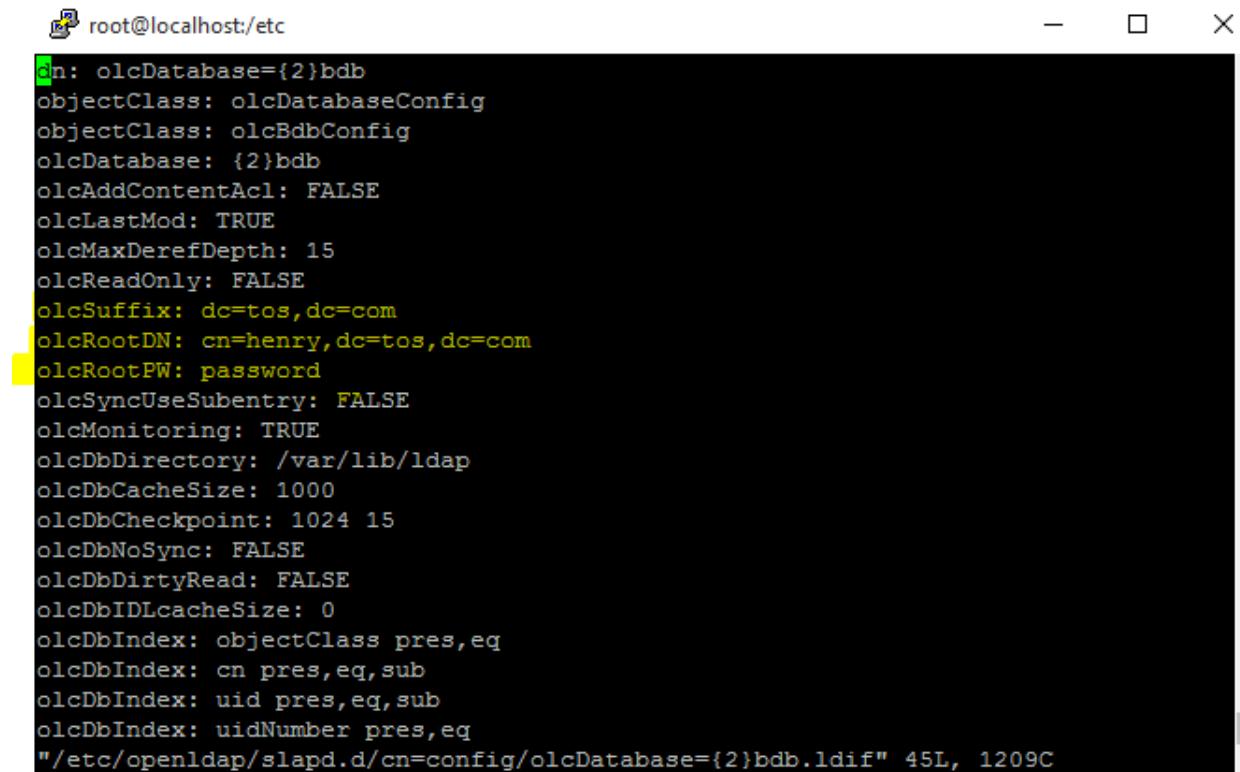
Modify the olcDatabase={2}hdb.ldif file, and change the olcRootDN entry. The following is the default entry.

```
# grep olcRootDN /etc/openldap/slapd.d/cn=config/olcDatabase={2}hdb.ldif
[root@tos Software]# grep olcRootDN /etc/openldap/slapd.d/cn\=config\olcDatabase
\={2\}hdb.ldif
olcRootDN: cn=Manager,dc=my-domain,dc=com
[root@tos Software]# 
```

Change the above line to an admin user. In this example, user “henry” will be the olcRootDN. Now setup the olcSuffix and to set the domain that you want

```
vi /etc/openldap/slapd.d/cn=config/olcDatabase={2}hdb.ldif
```

```
olcRootDN: cn=henry,dc=tos,dc=com
olcSuffix: dc=tos,dc=com
olcRootPW: password
```



```
root@localhost:/etc
[...]
olcDatabase={2}bdb
objectClass: olcDatabaseConfig
objectClass: olcBdbConfig
olcDatabase: {2}bdb
olcAddContentAcl: FALSE
olcLastMod: TRUE
olcMaxDerefDepth: 15
olcReadOnly: FALSE
olcSuffix: dc=tos,dc=com
olcRootDN: cn=henry,dc=tos,dc=com
olcRootPW: password
olcSyncUseSubentry: FALSE
olcMonitoring: TRUE
olcDbDirectory: /var/lib/ldap
olcDbCacheSize: 1000
olcDbCheckpoint: 1024 15
olcDbNoSync: FALSE
olcDbDirtyRead: FALSE
olcDbIDLcacheSize: 0
olcDbIndex: objectClass pres,eq
olcDbIndex: cn pres,eq,sub
olcDbIndex: uid pres,eq,sub
olcDbIndex: uidNumber pres,eq
"/etc/openldap/slapd.d/cn=config/olcDatabase={2}bdb.ldif" 45L, 1209C
```

The suffix line names the domain for which the LDAP server provides information and should be changed.

The rootdn entry is the Distinguished Name (DN) for a user who is unrestricted by access controls or administrative limit parameters set for operations on the LDAP directory. The rootdn user can be thought of as the root user for the LDAP directory.

### Verify the Configuration Files

Use slapttest command to verify the configuration file as shown below. This should display “testing succeeded” message as shown below.

```
slapttest -u
```

```
[root@localhost openldap]#
[root@localhost openldap]# slapttest -u
config file testing succeeded
[root@localhost openldap]#
```

Start **slapd** with the command:

```
service slapd start
```

```
[root@localhost ~]# service slapd start
Starting slapd:                                     [ OK ]
[root@localhost ~]#
```

To create these OU's, you can create an initial LDIF file as shown in the below example. In this example, this file allows you to create the base container which is dc=tos,dc=com and it creates two organizational units with the names users and groups in that container.

```
#cd /couchbase
```

Note: Create the folder if its not present [mkdir /couchbase]

```
# vi base.ldif
```

```
dn: dc=tos,dc=com
objectClass: dcObject
objectClass: organization
o: tos.com
dc: tos
```

```
dn: ou=users,dc=tos,dc=com
objectClass: organizationalUnit
objectClass: top
ou: users
```

```
dn: ou=groups,dc=tos,dc=com
objectClass: organizationalUnit
objectClass: top
ou: groups
```

Note: Ensure to maintain new line in between each dn entries.

```
[root@localhost couchbase]# more base.ldif
dn: dc=tos,dc=com
objectClass: dcObject
objectClass: organization
o: tos.com
dc: tos

dn: ou=users,dc=tos,dc=com
objectClass: organizationalUnit
objectClass: top
ou: users

dn: ou=groups,dc=tos,dc=com
objectClass: organizationalUnit
objectClass: top
ou: groups

[root@localhost couchbase]#
```

Import Base Structure Using ldapadd

Now we can import the base structure in to the LDAP directory using the ldapadd command as shown below. password should be password.

```
# ldapadd -x -W -D "cn=henry,dc=tos,dc=com" -f base.ldif
```

```
[root@localhost couchbase]# ldapadd -x -W -D "cn=henry,dc=tos,dc=com" -f base.ldif
Enter LDAP Password:
adding new entry "dc=tos,dc=com"

adding new entry "ou=users,dc=tos,dc=com"

adding new entry "ou=groups,dc=tos,dc=com"

[root@localhost couchbase]#
```

#### Verify the Base Structure using ldapsearch

To verify the OUs are successfully created, use the following ldapsearch command. Specify the password, which was entered in the ldif file database

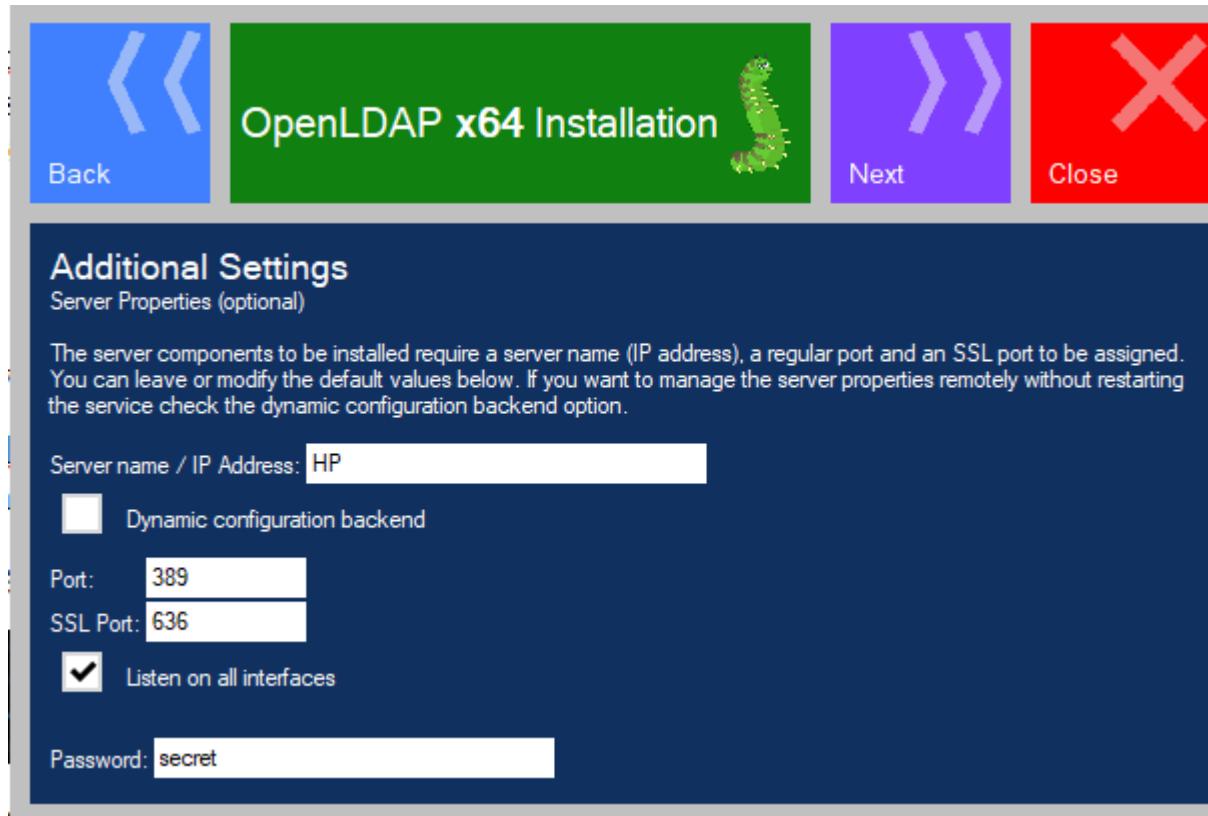
```
#ldapsearch -x -W -D "cn=henry,dc=tos,dc=com" -b "dc=tos,dc=com" "(objectclass=*)"
```

```
[root@localhost couchbase]# ldapsearch -x -W -D "cn=henry,dc=tos,dc=com" -b "dc=tos,dc=com" "(objectclass=*)"
Enter LDAP Password:
# extended LDIF
#
# LDAPv3
# base <dc=tos,dc=com> with scope subtree
# filter: (objectclass=*)
# requesting: ALL
#
# tos.com
dn: dc=tos,dc=com
objectClass: dcObject
objectClass: organization
o: tos.com
dc: tos

# users, tos.com
dn: ou=users,dc=tos,dc=com
objectClass: organizationalUnit
objectClass: top
```

## Windows

To install Open Ldap in your laptop/desktop double click OpenLDAPforWindows\_x64.exe and accepts all the default value except the following options.  
<https://www.userbooster.de/en/support/feature-articles.aspx>



It is always recommended to create a dedicated user account first with the full permissions to change information on the LDAP database.

Ensure that the following service is up:

	The Offline ...	Manual (Trig...	Local Syste...
	This service ...	Running	Automatic (D...
Offline Files		Manual	Local System
OneSyncSvc_795c3	Running	Automatic (D...	Local System
<b>OpenLDAP Service</b>	Running	Manual	Local System
Optimize drives	Helps the c...	Manual	Local System
Payments and NFC/SE Manager	Manages pa...	Running	Manual (Trig...

In this example, user "henry" will be the olcRootDN.

Now setup the olcSuffix and to set the domain that you want on the following configuration file.

D:\MyExperiment\OpenLDAP\ slapd.conf

```

29 #####
30 # mdb database definitions
31 #####
32
33
34 database      mdb
35 suffix        "dc=tos,dc=com"
36 rootdn        "cn=henry,dc=tos,dc=com"
37 # Cleartext passwords, especially for the rootdn, should
38 # be avoided. See slappasswd(8) and slapd.conf(5) for details.
39 # Use of strong authentication encouraged.
40 rootpw        {SSHA}eqOHOIXky1/5xok2d9XVKEqtE+ZWIaYH
41
42 # The database directory MUST exist prior to running slapd AND
43 # should only be accessible by the slapd and slap tools.
44 # Mode 700 recommended.
45 directory     ./data
46 searchstack 20
47 # Indices to maintain
48 index mail pres,eq
49 index objectclass pres
50 index default eq,sub
51 index sn eq,sub,subinitial
52 index telephonenumber
53 index cn
54

```

To create these OU's, you can create an initial LDIF file as shown in the below example. In this example, this file allows you to create the base container which is dc=tos,dc=com and it creates two organizational units with the names users and groups in that container.

#cd D:\

Note: Create a file base.ldif with the following entry in the d drive.

```

dn: dc=tos,dc=com
objectClass: dcObject
objectClass: organization
o: tos.com

```

dc: tos

```
dn: ou=users,dc=tos,dc=com
objectClass: organizationalUnit
objectClass: top
ou: users
```

```
dn: ou=groups,dc=tos,dc=com
objectClass: organizationalUnit
objectClass: top
ou: groups
```

Note: Ensure to maintain new line in between each dn entries.

```
[root@localhost couchbase]# more base.ldif
dn: dc=tos,dc=com
objectClass: dcObject
objectClass: organization
o: tos.com
dc: tos

dn: ou=users,dc=tos,dc=com
objectClass: organizationalUnit
objectClass: top
ou: users

dn: ou=groups,dc=tos,dc=com
objectClass: organizationalUnit
objectClass: top
ou: groups

[root@localhost couchbase]#
```

Import Base Structure Using ldapmodify

Now we can import the base structure in to the LDAP directory using the ldapadd command as shown below. password should be **secret**.

Open a command prompt and go to the OpenLDAP Installation folder.

```
#cd D:\MyExperiment\OpenLDAP\ClientTools>
```

Import Base Structure Using ldapadd

Now we can import the base structure in to the LDAP directory using the ldapadd command as shown below password should be secret.

```
# ldapmodify -a -x -W -D "cn=henry,dc=tos,dc=com" -f d:\base.ldif
```

```
D:\MyExperiment\OpenLDAP\ClientTools>ldapmodify.exe -a -x -W -D "cn=henry,dc=tos,dc=com" -f d:\base.ldif
Enter LDAP Password: secret
ldap_connect_to_host: TCP localhost:389
ldap_new_socket: 820
ldap_prepare_socket: 820
ldap_connect_to_host: Trying ::1 389
ldap_pvt_connect: fd: 820 tm: -1 async: 0
attempting to connect:
connect success
adding new entry "dc=tos,dc=com"
adding new entry "ou=users,dc=tos,dc=com"
adding new entry "ou=groups,dc=tos,dc=com"
```

Verify the Base Structure using ldapsearch

To verify the OUs are successfully created, use the following ldapsearch command. Specify the password, which was entered in the ldif file database. You need to execute the following command from the client tools folder of OpenLdap installation folder.

```
#ldapsearch -x -W -D "cn=henry,dc=tos,dc=com" -b "dc=tos,dc=com" "(objectclass=*)"
```

```
D:\MyExperiment\OpenLDAP\ClientTools>ldapsearch -x -W -D "cn=henry,dc=tos,dc=com" -b "dc=tos,dc=com" "(objectclass=*)"
Enter LDAP Password: secret
ldap_connect_to_host: TCP localhost:389
ldap_new_socket: 812
ldap_prepare_socket: 812
ldap_connect_to_host: Trying ::1 389
ldap_pvt_connect: fd: 812 tm: -1 async: 0
attempting to connect:
connect success
# extended LDIF
#
# LDAPv3
# base <dc=tos,dc=com> with scope subtree
# filter: (objectclass=*)
# requesting: ALL
#
# tos.com
dn: dc=tos,dc=com
objectClass: dcObject
objectClass: organization
o: tos.com
dc: tos
```

Congrats! You have installed and configured OpenLdap successfully.

Now you can proceed to the Security Lab.

----- Lab Ends Here -----

Notes:

Yellow → Verified in Version 6.0