CS300 Couchbase NoSQL Server Administration

Lab 6 Exercise Manual

Couchbase Learning Services

Release: 4.5

Revised: July 26th, 2016

Lab #6: XDCR

Objective: This 2 hour lab will walk you through the fundamentals of XDCR and teach you the core concepts behind Couchbase Cross Data Center Replication. This is a long lab, so please perform each step carefully as missing steps may cause unpredictable behavior!

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

- Connect to and create a 2-node remote Couchbase cluster. We will call this the "London" or Production/Source cluster. (The existing 6-node cluster will be called the "NYC" or Remote/Destination cluster.)
- Configure unencrypted, unidirectional replication between the 6-node and 2-node cluster for the beer-sample bucket.
- Use cbworkloadgen to push 250,000 keys from NYC (6-node) -> London (2-node)
- Learn that Design Docs and Views do not get replicated by default
- Learn about XDCR conflict resolution and learn how to get the revision count of a specific item from the data file
- See what a tombstone looks like in the data file and learn about tombstones
- Learn differences between version 1 (REST) and version 2 (memached REST) of the XDCR protocol
- Learn about advanced XDCR settings
- Viewing outbound and inbound replication statistics using the Web UI
- Viewing internal XDCR settings via the REST API
- Configure an encrypted, bidirectional replication between 2 clusters
- Learn about the optimistic threshold setting for replication streams
- Learn how to delete replication streams

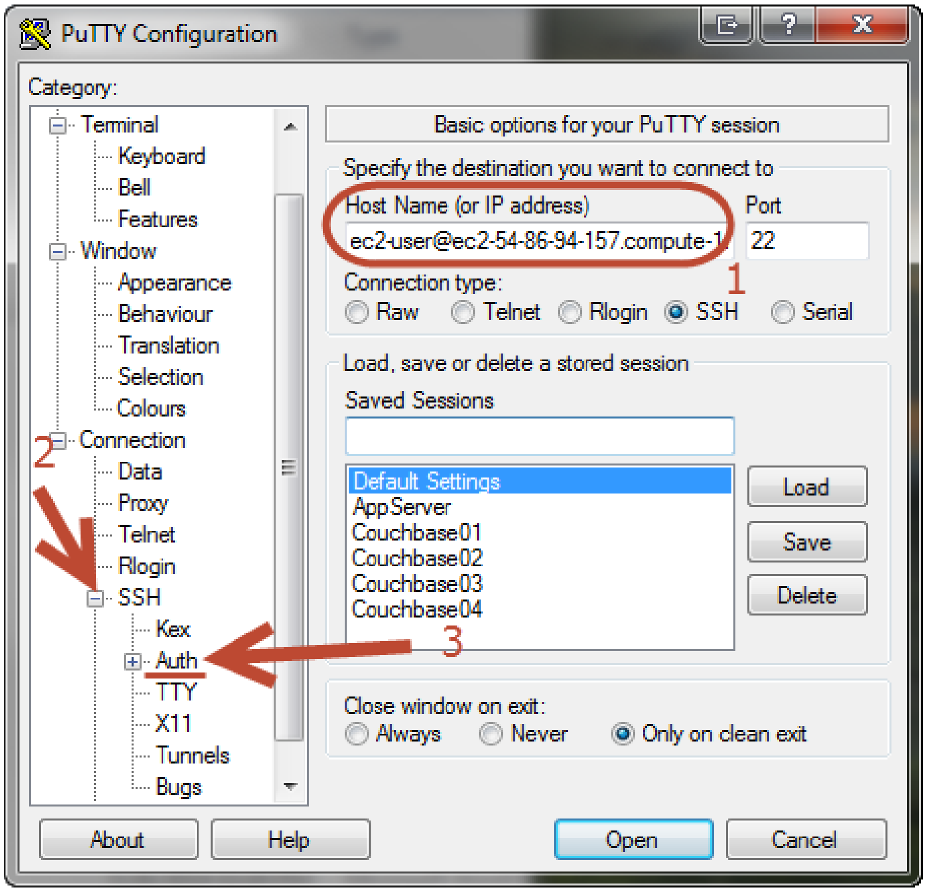
Create a 2-node cluster on the Remote side:

Before we configure Cross Data Center Replication (XDCR) we need to first set up a 2nd cluster in a remote data center. If you check the updated `Cluster-IPs` spreadsheet, the instructor will have provided you with 2 additional Virtual Machines in the Amazon cloud. Assume that these 2 new hosts are in a different datacenter than the existing 6-node cluster and we want to set up different types of replication streams for different buckets.

But first, let's quickly install Couchbase on the 2 remote nodes and configure them into a 2-node cluster.

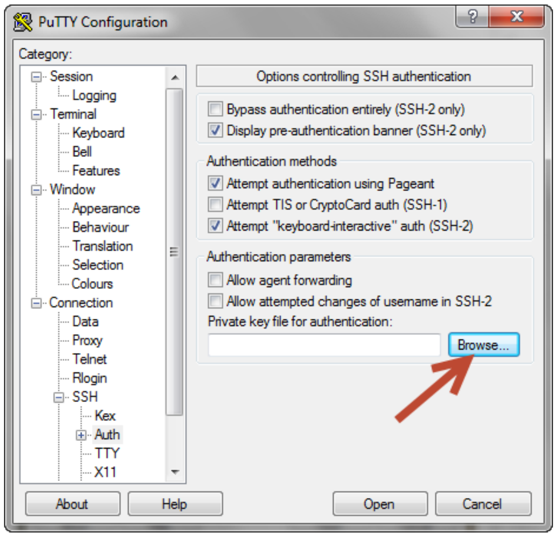
We'll need to create 2 more connections in PuTTY to connect to the 2 new servers.

Launch PuTTY and **type ec2-user@<the public hostname>** that the instructor gave you for Node #1 (**Red**) in the remote cluster into PuTTY and then **click on the + next to SSH** to expand its options and finally **select Auth**:



Host Name (or IP address): ec2-user@ec2-54-86-94-157.compute-1
Port: 22
Connection type: Raw Telnet Rlogin [✓ SSH] Serial
Close window on exit: Always Never [✓ Only on clean exit]

Click Browse to select the Private key file for authentication:



Options controlling SSH authentication

- [ ] Bypass authentication entirely (SSH-2 only)
- [✓] Display pre-authentication banner (SSH-2 only)

Authentication methods

- [✓] Attempt authentication using Pageant
- [ ] Attempt TIS or CryptoCard auth (SSH-1)
- [✓] Attempt "keyboard-interactive" auth (SSH-2)

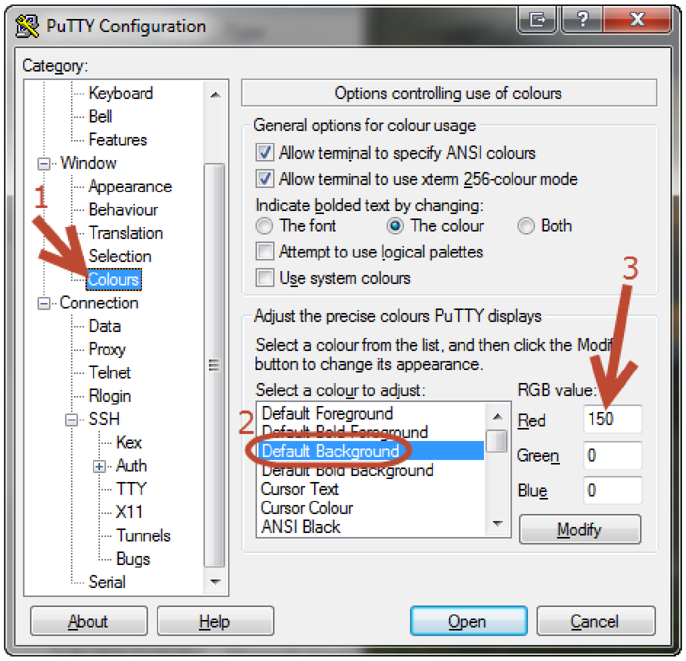
Authentication parameters

- [ ] Allow agent forwarding
- [ ] Allow attempted changes of username in SSH-2

Private key file for authentication:

Choose the **"Amazon-Private-Key.ppk"** file that the instructor provided you with.

In the left pane, **click on Colors**, then under "Select a colour to adjust" **choose Default Background** and alter the **Red RGB value to 150**.



Options controlling use of colours

General options for colour usage

- [✓] Allow terminal to specify ANSI colours
- [✓] Allow terminal to use xterm 256-colour mode

Indicate bolded text by changing:
( ) The font [✓ The colour] ( ) Both

- [ ] Attempt to use logical palettes
- [ ] Use system colours

Adjust the precise colours PuTTY displays

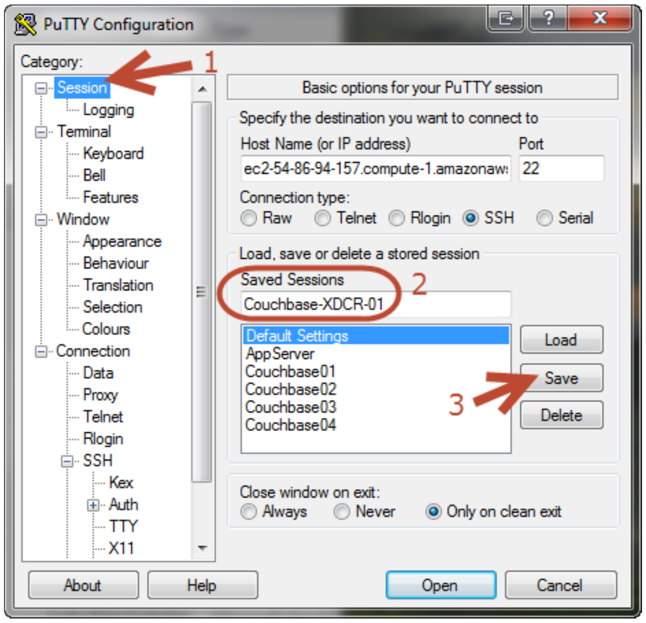
Select a colour from the list, and then click the Modi button to change its appearance.

Select a colour to adjust:
Default Foreground
Default Bold Foreground
[✓ Default Background]
Default Bold Background
Cursor Text
Cursor Colour
ANSI Black

RGB value:

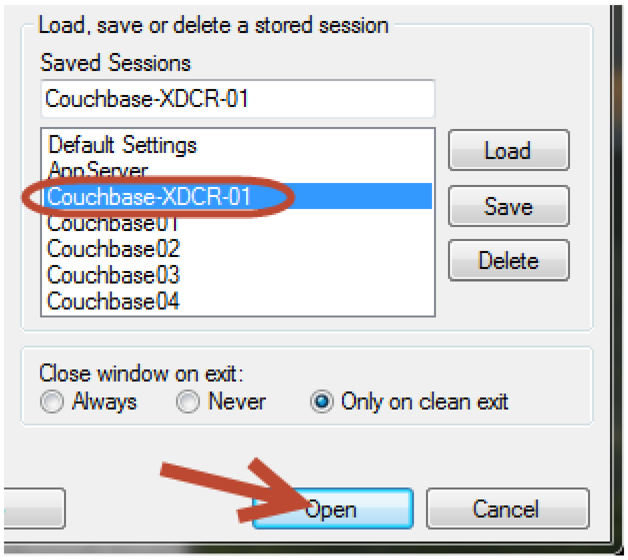
Red	150
Green	0
Blue	0

Next, **click on Session** and type to **save the session as "Couchbase-XDCR-01"**, Then **click on Save**. For example, here the session is being saved as "Couchbase-XDCR-01":



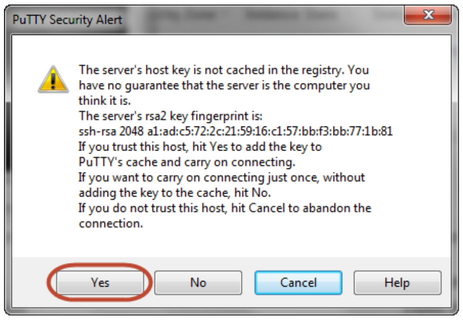
Specify the destination you want to connect to
Host Name (or IP address): ec2-54-86-94-157.compute-1.amazonaw:
Port: 22
Connection type: Raw Telnet Rlogin [✓ SSH] Serial
Load, save or delete a stored session
Saved Sessions: Couchbase-XDCR-01
Default Settings
AppServer
Couchbase01
Couchbase02
Couchbase03
Couchbase04
Close window on exit: Always Never [✓ Only on clean exit]

Now highlight **Couchbase-XDCR-01** and click **Open** to connect to this VM:



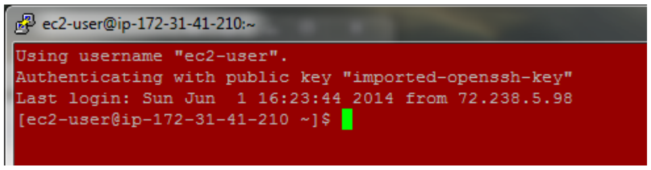
Load, save or delete a stored session
Saved Sessions: Couchbase-XDCR-01
Default Settings
AppServer
Couchbase-XDCR-01
Couchbase01
Couchbase02
Couchbase03
Couchbase04
Close window on exit: Always Never [✓ Only on clean exit]

You will have to click **"Yes"** to a message about the server's rsa2 key before a successful connection.



The server's host key is not cached in the registry. You have no guarantee that the server is the computer you think it is.
The server's rsa2 key fingerprint is:
ssh-rsa 2048 a1:ad:c5:72:2c:21:59:16:c1:57:bb:f3:bb:77:1b:81
If you trust this host, hit Yes to add the key to PuTTY's cache and carry on connecting.
If you want to carry on connecting just once, without adding the key to the cache, hit No.
If you do not trust this host, hit Cancel to abandon the connection.

You should now be logged into the first remote server:



```
Using username "ec2-user".
Authenticating with public key "imported-openssh-key"
Last login: Sun Jun  1 16:23:44 2014 from 72.238.5.98
[ec2-user@ip-172-31-41-210 ~]$
```

Next we will:

- Turn off the linux firewall
- Disable swappiness
- Install Couchbase Server

Turn off the Linux firewalld(may not be needed in amazon cloud)

- #**systemctl stop firewalld**
```
Failed to issue method call: Unit firewalld.service not loaded.
```

Note, you should not turn off the firewall like this in a production Couchbase cluster! In that case you should instead refer to this URL to see which ports need to be selectively opened up for Couchbase:

Turn off swapping for the running system, but first switch to root user:

```
[ec2-user@ip-172-31-19-30 ~]$ sudo -s
[root@ip-172-31-19-30 ec2-user]# echo 0 > /proc/sys/vm/swappiness
```

Then permanently make this change in the sysctl.conf file, so the change persists after a reboot and exit root:

```
[root@ip-172-31-19-30 ec2-user]# echo '' >> /etc/sysctl.conf
[root@ip-172-31-19-30 ec2-user]# echo '#Set swappiness to 0 to avoid
swapping' >> /etc/sysctl.conf
[root@ip-172-31-19-30 ec2-user]# echo 'vm.swappiness = 0' >>
/etc/sysctl.conf
```

Disable Transparent Huge Pages

In a production Couchbase cluster, it is very important to disable Transparent Huge pages on each node.

```
# Disable THP on a running system
echo never > /sys/kernel/mm/transparent_hugepage/enabled
echo never > /sys/kernel/mm/transparent_hugepage/defrag
```

Install Couchbase:

```
# yum install wget
Loaded plugins: amazon-id, rhui-lb
rhui-REGION-client-config-server-7                                 | 2.9 kB  00
rhui-REGION-rhel-server-releases                                   | 3.7 kB  00
rhui-REGION-rhel-server-rh-common                                  | 1.9 kB  00
(1/4): rhui-REGION-client-config-server-7/x86_64/primary_db        | 5.0 kB  00
(2/4): rhui-REGION-rhel-server-rh-common/7Server/x86_64/primary    |  30 kB  00
(3/4): rhui-REGION-rhel-server-rh-common/7Server/x86_64/updateinfo |  11 kB  00
(4/4): rhui-REGION-rhel-server-releases/7Server/x86_64/primary_db  |  13 MB  00
(1/2): rhui-REGION-rhel-server-releases/7Server/x86_64/group_gz    | 133 kB  00
(2/2): rhui-REGION-rhel-server-releases/7Server/x86_64/updateinfo  | 590 kB  00
rhui-REGION-rhel-server-rh-common
Resolving Dependencies
--> Running transaction check
---> Package wget.x86_64 0:1.14-10.el7_0.1 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

=================================================================================================
 Package                Arch                    Version                          Repository
=================================================================================================
Installing:
 wget                   x86_64                  1.14-10.el7_0.1                  rhui-REGION-rhel-
server-releases

Transaction Summary
=================================================================================================
Install  1 Package

Total download size: 546 k
Installed size: 2.0 M
Is this ok [y/d/N]: y
Downloading packages:
wget-1.14-10.el7_0.1.x86_64.rpm
| 546 kB  00
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : wget-1.14-10.el7_0.1.x86_64
  Verifying  : wget-1.14-10.el7_0.1.x86_64

Installed:
  wget.x86_64 0:1.14-10.el7_0.1

Complete!
```

```
# exit
```

Download Couchbase 4.5.0 EE:

```
[ec2-user@ip-172-31-19-30 ~]$ wget
http://packages.couchbase.com/releases/4.5.0/couchbase-server-
enterprise-4.5.0-centos7.x86_64.rpm
--2014-10-22 17:36:19--  http://packages.couchbase.com/releases/4.5.0/couchbase-server-
enterprise-4.5.0-centos6.x86_64.rpm
Resolving packages.couchbase.com... 54.231.12.112
Connecting to packages.couchbase.com|54.231.12.112|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 205018340 (196M) [application/x-redhat-package-manager]
Saving to: “couchbase-server-enterprise-4.5.0-centos6.x86_64.rpm”

100%[===============================================================================================
==============================>] 205,018,340 1.65M/s   in 67s

2014-10-22 17:37:27 (2.93 MB/s) - “couchbase-server-enterprise-4.5.0-centos6.x86_64.rpm” saved
[205018340/205018340]
```

Install Couchbase (note, this command might take 1-2 minutes to complete):

```
[ec2-user@ip-172-31-19-30 ~]$ sudo rpm --install couchbase-server-
enterprise-4.5.0-centos7.x86_64.rpm
Minimum RAM required  : 4 GB
System RAM configured : 3.45 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.service.

You have successfully installed Couchbase Server.
Please browse to http://couch08:8091/ 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, 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.
```

Note: in Couchbase 4.1.0 autostart was dependent on RH7.1 symbolic links that are not present in RH7.2 Please use the following to start couchbase. (this will be remedied in Couchbase 4.1.1)

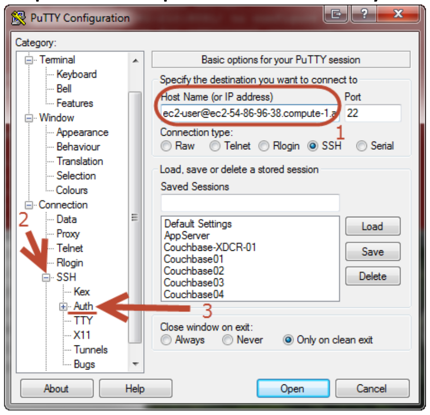
```
/opt/couchbase/etc/couchbase_init.d start
Starting couchbase-server

/opt/couchbase/etc/couchbase_init.d stop
Stopping couchbase-server

/opt/couchbase/etc/couchbase_init.d status
Obtaining system status
```

After the install finishes, wait 30 seconds, then check the status of the Couchbase Server:

```
[ec2-user@ip-172-31-25-69 ~]$ sudo /etc/init.d/couchbase-server status
couchbase-server is running
```

Next, we need to connect to and install Couchbase Server on the 2nd node in the Remote data center.

Launch PuTTY and **type ec2-user@<the public hostname>** that the instructor gave you for Node #2 (**Orange**) in the remote cluster into PuTTY and then **click on the + next to SSH** to expand its options and finally **select Auth**:



Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address): ec2-user@ec2-54-86-96-38.compute-1.a

Port: 22

Connection type: Raw Telnet Rlogin [✓ SSH] Serial

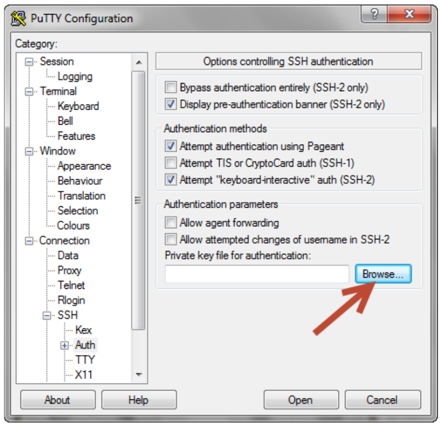
Load, save or delete a stored session

Saved Sessions

Default Settings
AppServer
Couchbase-XDCR-01
Couchbase01
Couchbase02
Couchbase03
Couchbase04

Close window on exit: Always Never [✓ Only on clean exit]

Click Browse to select the Private key file for authentication:



Options controlling SSH authentication

- [ ] Bypass authentication entirely (SSH-2 only)
- [✓] Display pre-authentication banner (SSH-2 only)

Authentication methods

- [✓] Attempt authentication using Pageant
- [ ] Attempt TIS or CryptoCard auth (SSH-1)
- [✓] Attempt "keyboard-interactive" auth (SSH-2)

Authentication parameters

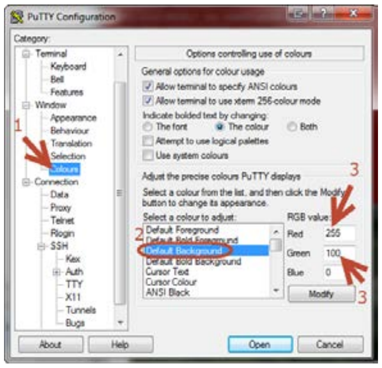
- [ ] Allow agent forwarding
- [ ] Allow attempted changes of username in SSH-2

Private key file for authentication:

Choose the **"Amazon-Private-Key.ppk"** file that the instructor provided you with.

In the left pane, **click on Colors**, then under "Select a colour to adjust" **choose Default Background** and alter the **Red RGB value to 255 and the Green RGB value to 100**.

Couchbase



Options controlling use of colours

General options for colour usage
- [✓] Allow terminal to specify ANSI colours
- [✓] Allow terminal to use xterm 256-colour mode

Indicate bolded text by changing: The font [✓ The colour] Both
- [ ] Attempt to use logical palettes
- [ ] Use system colours

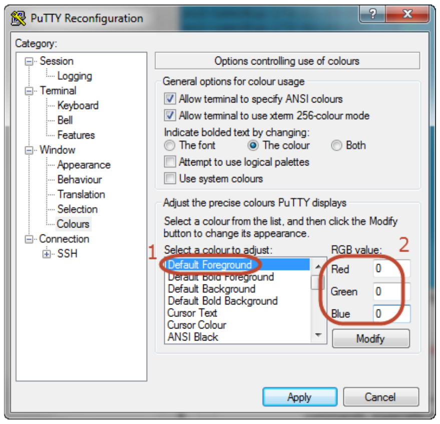
Adjust the precise colours PuTTY displays

Select a colour from the list, and then click the Modify button to change its appearance.

Select a colour to adjust: Default Foreground, Default Bold Foreground, [✓ Default Background], Default Bold Background, Cursor Text, Cursor Colour, ANSI Black

RGB value: Red 255, Green 100, Blue 0

In the same window **choose Default Foreground** under “Select a colour to adjust” and alter the **Red, Green and Blue values to 0** to make the text color black.



Options controlling use of colours

General options for colour usage
- [✓] Allow terminal to specify ANSI colours
- [✓] Allow terminal to use xterm 256-colour mode

Indicate bolded text by changing: The font [✓ The colour] Both
- [ ] Attempt to use logical palettes
- [ ] Use system colours

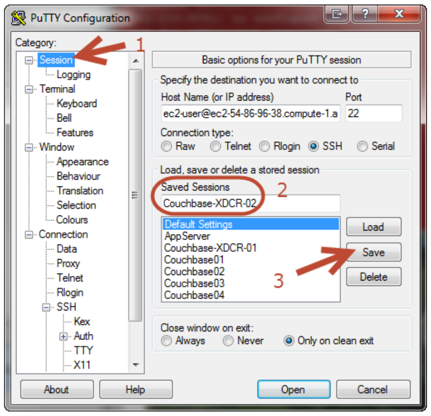
Adjust the precise colours PuTTY displays

Select a colour from the list, and then click the Modify button to change its appearance.

Select a colour to adjust: [✓ Default Foreground], Default Bold Foreground, Default Background, Default Bold Background, Cursor Text, Cursor Colour, ANSI Black

RGB value: Red 0, Green 0, Blue 0

Next, **click on Session** and type to **save the session as “Couchbase-XDCR-02”**, Then **click on Save**. For example, here the session is being saved as “Couchbase-XDCR-02”:



Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address): ec2-user@ec2-54-86-96-38.compute-1.a Port: 22

Connection type: Raw Telnet Rlogin [✓ SSH] Serial

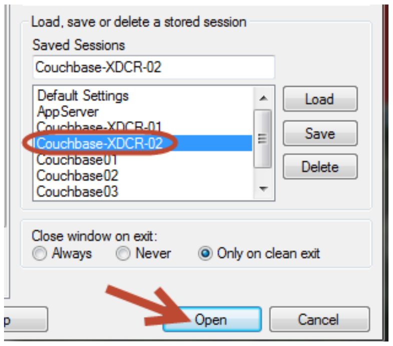
Load, save or delete a stored session

Saved Sessions: Couchbase-XDCR-02

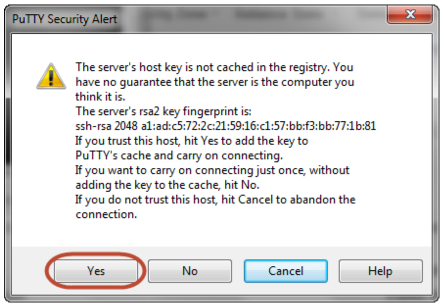
Default Settings
AppServer
Couchbase-XDCR-01
Couchbase01
Couchbase02
Couchbase03
Couchbase04

Close window on exit: Always Never [✓ Only on clean exit]

Now highlight **Couchbase-XDCR-02** and click **Open** to connect to this VM:

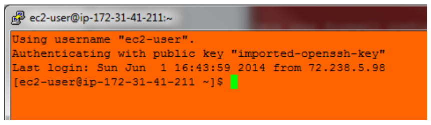


You will have to click **"Yes"** to a message about the server's rsa2 key before a successful connection.



The server's host key is not cached in the registry. You have no guarantee that the server is the computer you think it is.
The server's rsa2 key fingerprint is:
ssh-rsa 2048 a1:ad:c5:72:2c:21:59:16:c1:57:bb:f3:bb:77:1b:81
If you trust this host, hit Yes to add the key to PuTTY's cache and carry on connecting.
If you want to carry on connecting just once, without adding the key to the cache, hit No.
If you do not trust this host, hit Cancel to abandon the connection.

You should now be logged into the second remote server:



```
Using username "ec2-user".
Authenticating with public key "imported-openssh-key"
Last login: Sun Jun  1 16:43:59 2014 from 72.238.5.98
[ec2-user@ip-172-31-41-211 ~]$
```

Next we will:

- Turn off the linux firewall
- Disable swappiness
- Install Couchbase Server

Turn off the Linux firewalld(may not be needed in amazon cloud)

- `#systemctl stop firewalld`

```
Failed to issue method call: Unit firewalld.service not
loaded.
```

Note, you should not turn off the firewall like this in a production Couchbase cluster! In that case you should instead refer to this URL to see which ports need to be selectively opened up for Couchbase:

Turn off swapping for the running system, but first switch to root user:

```
[ec2-user@ip-172-31-19-30 ~]$ sudo -s
[root@ip-172-31-19-30 ec2-user]# echo 0 > /proc/sys/vm/swappiness
```

Then permanently make this change in the sysctl.conf file, so the change persists after a reboot and exit root:

```
[root@ip-172-31-19-30 ec2-user]# echo '' >> /etc/sysctl.conf
[root@ip-172-31-19-30 ec2-user]# echo '#Set swappiness to 0 to avoid
swapping' >> /etc/sysctl.conf
[root@ip-172-31-19-30 ec2-user]# echo 'vm.swappiness = 0' >>
/etc/sysctl.conf
```

Disable Transparent Huge Pages

In a production Couchbase cluster, it is very important to disable Transparent Huge pages on each node.

```
# Disable THP on a running system
echo never > /sys/kernel/mm/transparent_hugepage/enabled
echo never > /sys/kernel/mm/transparent_hugepage/defrag
```

Install Couchbase:

```
# yum install wget
Loaded plugins: amazon-id, rhui-lb
rhui-REGION-client-config-server-7                               | 2.9 kB  00
rhui-REGION-rhel-server-releases                                 | 3.7 kB  00
rhui-REGION-rhel-server-rh-common                                | 1.9 kB  00
(1/4): rhui-REGION-client-config-server-7/x86_64/primary_db      | 5.0 kB  00
(2/4): rhui-REGION-rhel-server-rh-common/7Server/x86_64/primary  |  30 kB  00
(3/4): rhui-REGION-rhel-server-rh-common/7Server/x86_64/updateinfo |  11 kB  00
(4/4): rhui-REGION-rhel-server-releases/7Server/x86_64/primary_db |  13 MB  00
(1/2): rhui-REGION-rhel-server-releases/7Server/x86_64/group_gz  | 133 kB  00
(2/2): rhui-REGION-rhel-server-releases/7Server/x86_64/updateinfo | 590 kB  00
rhui-REGION-rhel-server-rh-common
Resolving Dependencies
--> Running transaction check
---> Package wget.x86_64 0:1.14-10.el7_0.1 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

==================================================================================================
 Package                Arch                   Version                           Repository
==================================================================================================
Installing:
 wget                   x86_64                 1.14-10.el7_0.1                   rhui-REGION-rhel-
server-releases

Transaction Summary
==================================================================================================
Install  1 Package
```

```
Total download size: 546 k
Installed size: 2.0 M
Is this ok [y/d/N]: y
Downloading packages:
wget-1.14-10.el7_0.1.x86_64.rpm
| 546 kB  00
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : wget-1.14-10.el7_0.1.x86_64
  Verifying  : wget-1.14-10.el7_0.1.x86_64

Installed:
  wget.x86_64 0:1.14-10.el7_0.1

Complete!

# exit
```

Download Couchbase 4.5.0 EE:

```
[ec2-user@ip-172-31-19-30 ~]$
wget http://packages.couchbase.com/releases/4.5.0/couchbase-server-enterprise-4.5.0-centos7.x86_64.rpm
--2014-10-22 17:36:24--  http://packages.couchbase.com/releases/4.5.0/couchbase-server-
enterprise-4.5.0-centos6.x86_64.rpm
Resolving packages.couchbase.com... 205.251.243.139
Connecting to packages.couchbase.com|205.251.243.139|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 205018340 (196M) [application/x-redhat-package-manager]
Saving to: “couchbase-server-enterprise-4.5.0-centos6.x86_64.rpm”

100%[============================================================>] 205,018,340 2.00M/s   in 68s

2014-10-22 17:37:33 (2.86 MB/s) - “couchbase-server-enterprise-4.5.0-centos6.x86_64.rpm” saved
[205018340/205018340]
```

Install Couchbase (note, this command might take 1-2 minutes to complete):

```
[ec2-user@ip-172-31-19-30 ~]$ sudo rpm --install couchbase-server-enterprise-4.5.0-centos7.x86_64.rpm
Minimum RAM required  : 4 GB
System RAM configured : 3.45 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.service.

You have successfully installed Couchbase Server.
Please browse to http://couch08:8091/ 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, 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.
```

After the install finishes, wait 30 seconds, then check the status of the Couchbase Server:

```
[ec2-user@ip-172-31-25-69 ~]$ sudo /etc/init.d/couchbase-server status
couchbase-server is running
```

Note, that the color coding for all of the servers should now be:

App Server: Black

Cluster #1

- Node #1: Dark Blue
- Node #2: Light Blue
- Node #3: Green
- Node #4: Yellow
- Node #5 Pink
- Node #6 watermelon

Cluster #2

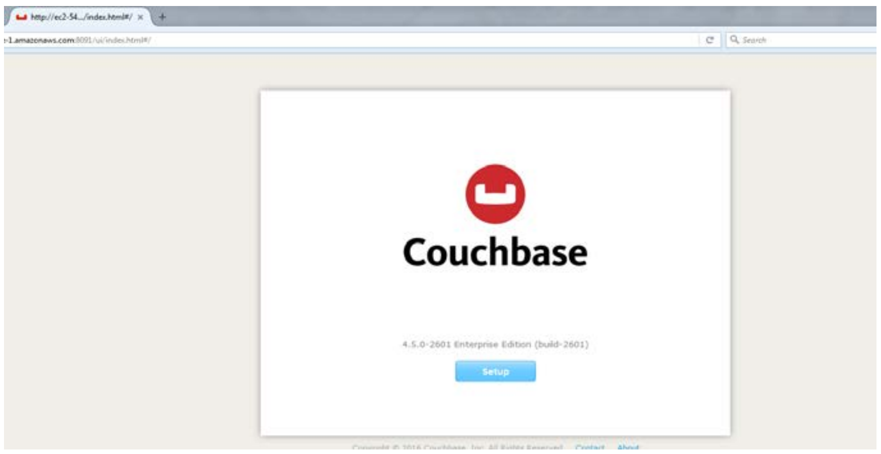
XDCR #1: Red

XDCR #2: Orange

Finally, let's configure a 2-node cluster on the remote side.

Open a new browser tab and connect to the XDCR #1 node (Red)'s setup wizard. Open a Chrome or Firefox browser and go to the following URL:

http://<public hostname of XDCR #1 VM>:8091


http://ec2-54.../index.html#/
-1.amazonaws.com:8091/ui/index.html#/
Couchbase
4.5.0-2601 Enterprise Edition (build-2601)
Setup

Click on SETUP in the bottom to continue.

Some of the settings on the "CONFIGURE SERVER" page will need to be altered. Specifically, the items in red need to be changed:

Databases Path: /opt/couchbase/var/lib/couchbase/data

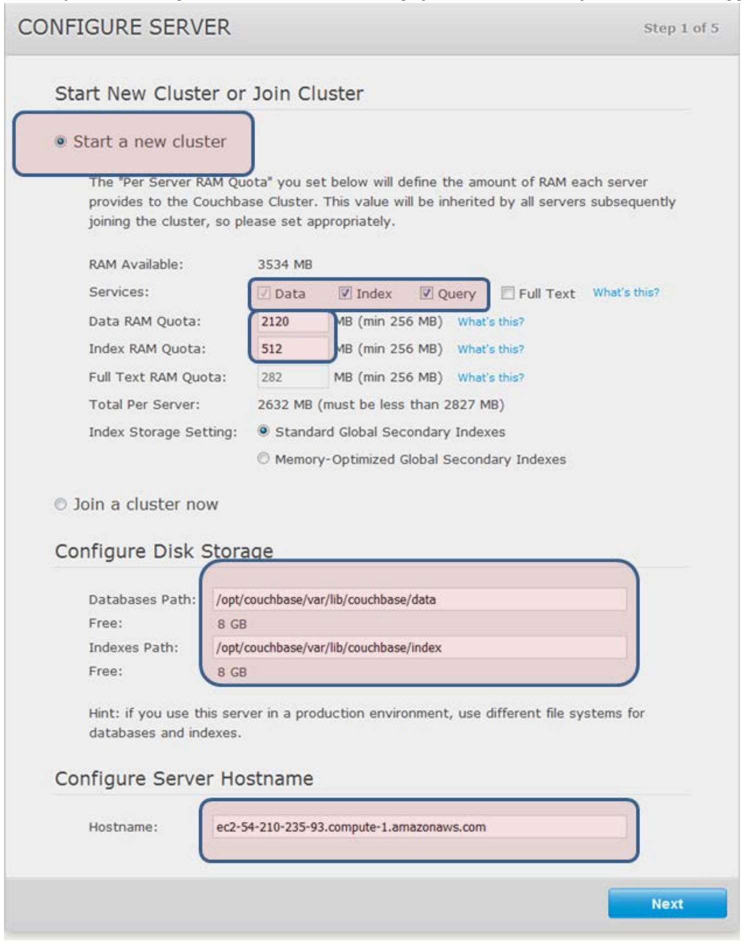
Indices Path: /opt/couchbase/var/lib/couchbase/**index**

Hostname: **<Public hostname of VM, retrieve this from the Cluster-IPs spreadsheet> Or from the browser URL entry line. i.e. <ec2-54-174-65-105.compute-1.amazonaws.com>**

Choose: Start a new cluster
Services select: **Data, Index & Query**

Per Server DATA RAM Quota: **2120 MB** *(the default is 60% of the node's total RAM) Accept the default calculation if your memory value is different than shown*

Per Server Index RAM Quota: **256 MB** *(Min ram value shown to right) Accept the default calculation if your memory value is different than shown.*


CONFIGURE SERVER Step 1 of 5

Start New Cluster or Join Cluster

◉ Start a new cluster

The "Per Server RAM Quota" you set below will define the amount of RAM each server provides to the Couchbase Cluster. This value will be inherited by all servers subsequently joining the cluster, so please set appropriately.

RAM Available:	3534 MB
Services:	☑ Data ☑ Index ☑ Query ☐ Full Text What's this?
Data RAM Quota:	2120 MB (min 256 MB) What's this?
Index RAM Quota:	512 MB (min 256 MB) What's this?
Full Text RAM Quota:	282 MB (min 256 MB) What's this?
Total Per Server:	2632 MB (must be less than 2827 MB)
Index Storage Setting:	◉ Standard Global Secondary Indexes ○ Memory-Optimized Global Secondary Indexes

○ Join a cluster now

Configure Disk Storage

Databases Path:	/opt/couchbase/var/lib/couchbase/data
Free:	8 GB
Indexes Path:	/opt/couchbase/var/lib/couchbase/index
Free:	8 GB

Hint: if you use this server in a production environment, use different file systems for databases and indexes.

Configure Server Hostname

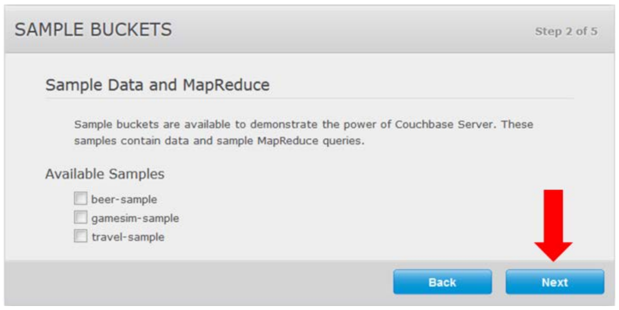
Hostname: ec2-54-210-235-93.compute-1.amazonaws.com

Next

Click on Next to continue:

Choose Data, Index and query for services. (all three to run on a single node)

Simply click on Next to continue (we do not want to load any sample buckets on this cluster at this time):


SAMPLE BUCKETS Step 2 of 5

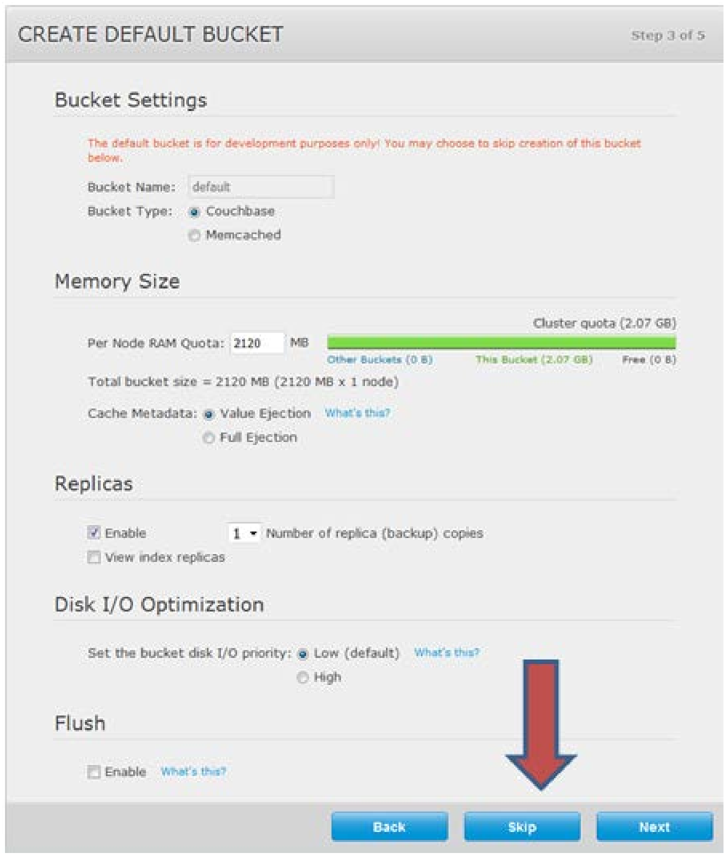
Sample Data and MapReduce

Sample buckets are available to demonstrate the power of Couchbase Server. These samples contain data and sample MapReduce queries.

Available Samples

- [ ] beer-sample
- [ ] gamesim-sample
- [ ] travel-sample

You may now skip the creation of the default bucket. just click Skip:


CREATE DEFAULT BUCKET Step 3 of 5

Bucket Settings

The default bucket is for development purposes only! You may choose to skip creation of this bucket below.

Bucket Name: default

Bucket Type: [✓ Couchbase] Memcached

Memory Size

Cluster quota (2.07 GB)

Per Node RAM Quota: 2120 MB

Other Buckets (0 B) This Bucket (2.07 GB) Free (0 B)

Total bucket size = 2120 MB (2120 MB x 1 node)

Cache Metadata: [✓ Value Ejection] Full Ejection What's this?

Replicas

- [x] Enable 1 Number of replica (backup) copies
- [ ] View index replicas

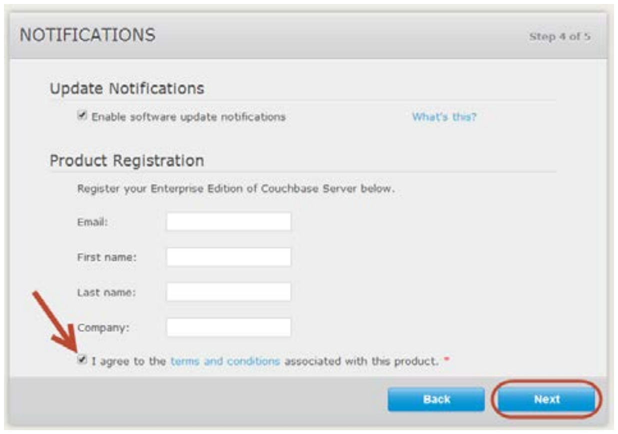
Disk I/O Optimization

Set the bucket disk I/O priority: [✓ Low (default)] High What's this?

Flush

- [ ] Enable What's this?

Place a check next to "I agree" and click Next:



NOTIFICATIONS Step 4 of 5

Update Notifications

☑ Enable software update notifications What's this?

Product Registration

Register your Enterprise Edition of Couchbase Server below.

Email:

First name:

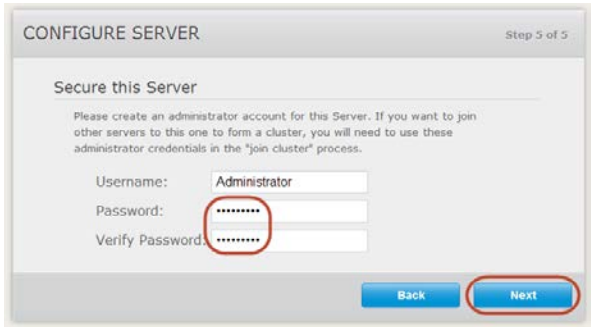
Last name:

Company:

☑ I agree to the terms and conditions associated with this product. *

Back Next

Type couchbase in lower case twice as the password and click Next:



CONFIGURE SERVER Step 5 of 5

Secure this Server

Please create an administrator account for this Server. If you want to join other servers to this one to form a cluster, you will need to use these administrator credentials in the "join cluster" process.

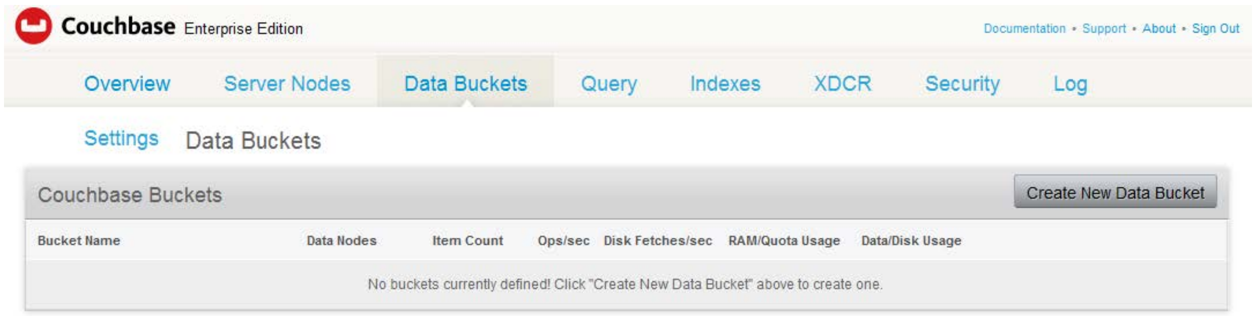
Username: Administrator

Password: ••••••••••

Verify Password: ••••••••••

Back Next

In a few moments, you will see the Couchbase cluster dashboard. Click on Data Buckets:



Couchbase Enterprise Edition

Documentation • Support • About • Sign Out

Overview Server Nodes Data Buckets Query Indexes XDCR Security Log

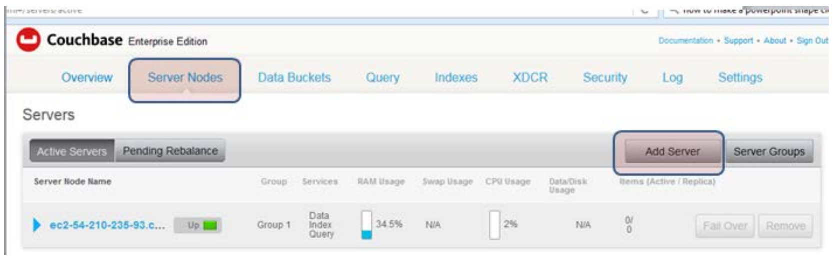
Settings Data Buckets

Couchbase Buckets Create New Data Bucket

Bucket Name	Data Nodes	Item Count	Ops/sec	Disk Fetches/sec	RAM/Quota Usage	Data/Disk Usage

No buckets currently defined! Click "Create New Data Bucket" above to create one.

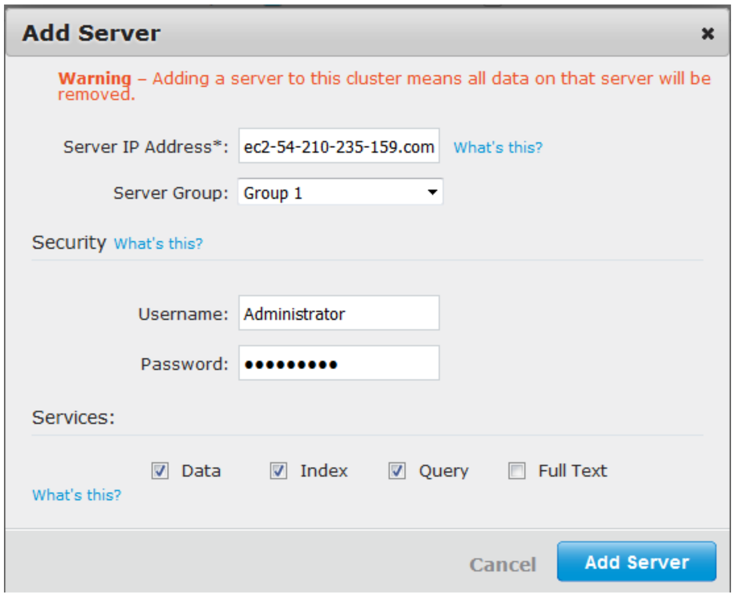
You should now see no buckets configured in the cluster:
Deleting the default bucket will make adding a 2nd node in this cluster faster as the rebalance operation will complete significantly faster.
Let's continue with adding the 2nd node into this remote cluster.
At the top of the page, click on Server Nodes and then 'Add Server' to grow this cluster:



Server Node Name	Group	Services	RAM Usage	Swap Usage	CPU Usage	Data/Disk Usage	Items (Active / Replica)
ec2-54-210-235-93.c... Up	Group 1	Data Index Query	34.5%	N/A	2%	N/A	0/ 0

On the Add server popup –

- **Type in the Server IP address of XDCR Node #2 (Orange VM) as the Server IP Address**
- **Select Group 1 for the server group**
- **Username should remain 'Administrator'**
- **Choose 'couchbase' for the password**
- **Choose Data, Index and query for services. (all three to run on a single node)**
- **Click Add Server**



Warning – Adding a server to this cluster means all data on that server will be removed.

Server IP Address*: ec2-54-210-235-159.com

Server Group: Group 1

Security

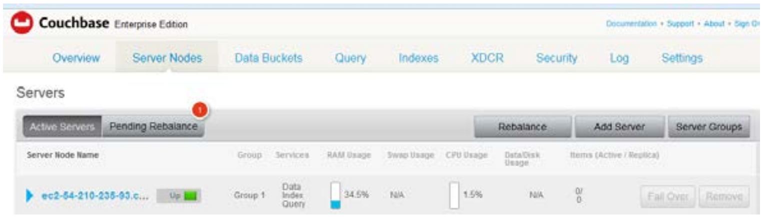
Username: Administrator

Password: ●●●●●●●●●

Services:

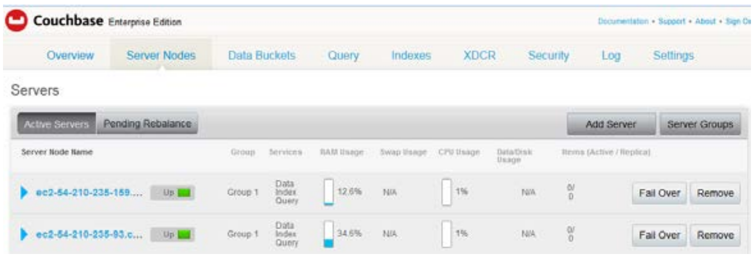
☑ Data ☑ Index ☑ Query ☐ Full Text

You should now be returned to the Web UI page for Server Nodes and see a Pending Rebalance. Click on Rebalance:



Server Node Name		Group	Services	RAM Usage	Swap Usage	CPU Usage	Data/Disk Usage	Items (Active / Replica)
ec2-54-210-235-93.c...	Up	Group 1	Data Index Query	34.5%	N/A	1.5%	N/A	0/ 0

Within a few seconds the Rebalance operation should complete and you will have a healthy 2-node cluster in the remote datacenter!



Server Node Name		Group	Services	RAM Usage	Swap Usage	CPU Usage	Data/Disk Usage	Items (Active / Replica)
ec2-54-210-235-159....	Up	Group 1	Data Index Query	12.6%	N/A	1%	N/A	0/ 0
ec2-54-210-235-93.c...	Up	Group 1	Data Index Query	34.6%	N/A	1%	N/A	0/ 0

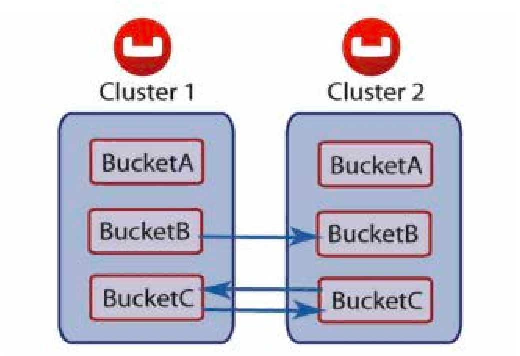
Configure unidirectional replication for the beer-sample bucket:

In the remainder of this lab, we will configure XDCR using different techniques and learn about some important replication settings.

First, let’s configure our first replication to replicate the beer-sample bucket from the production 6-node cluster to the remote 2-node cluster.

There are 2 replication topologies: unidirectional and bidirectional.

The diagram below shows BucketB -> BucketB configured with unidirectional replication. Also notice that BucketC is configured with bidirectional replication.



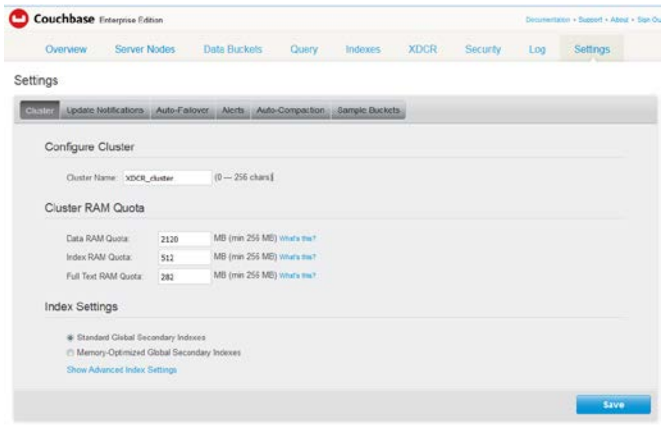
Unidirectional Replication is one-way replication, where active data gets replicated from the source cluster to the destination cluster. You may use unidirectional replication when you want to create an active offsite backup, replicating data from one cluster to a backup cluster.

Bidirectional Replication allows two clusters to replicate data with each other. Setting up bidirectional replication in Couchbase Server involves setting up two unidirectional replication links from one cluster to the other. This is useful when you want to load balance your workload across two clusters where each cluster bidirectionally replicates data to the other cluster.

XDCR can be setup on a per bucket basis. A bucket is a logical container for documents in Couchbase Server. Depending on your application requirements, you might want to replicate only a subset of the data in Couchbase Server between two clusters. With XDCR you can selectively pick which buckets to replicate between two clusters in a unidirectional or bidirectional fashion.

We will configure unidirectional replication for the beer-sample bucket.

Since couchbase 3.X we can name(label) our clusters. Click on Settings. And Cluster. And name your cluster. **XDCR Cluster**



Settings

Configure Cluster

Cluster Name: XDCR_cluster (0 — 256 chars)

Cluster RAM Quota

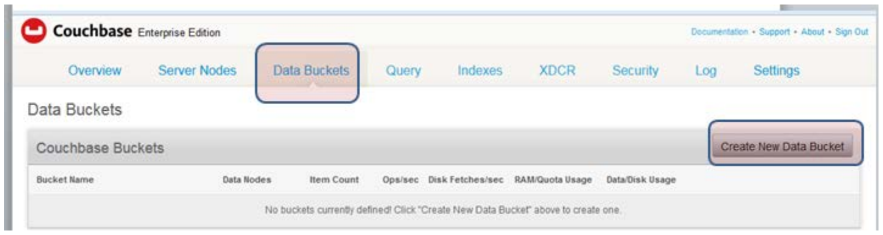
Data RAM Quota:	2120	MB (min 256 MB)
Index RAM Quota:	512	MB (min 256 MB)
Full Text RAM Quota:	282	MB (min 256 MB)

Index Settings

[✓ Standard Global Secondary Indexes]
Memory-Optimized Global Secondary Indexes

Before setting up replication, we need to create an empty beer-sample bucket in the remote data center's 2-node cluster. Replication will not automatically create the destination bucket for you.

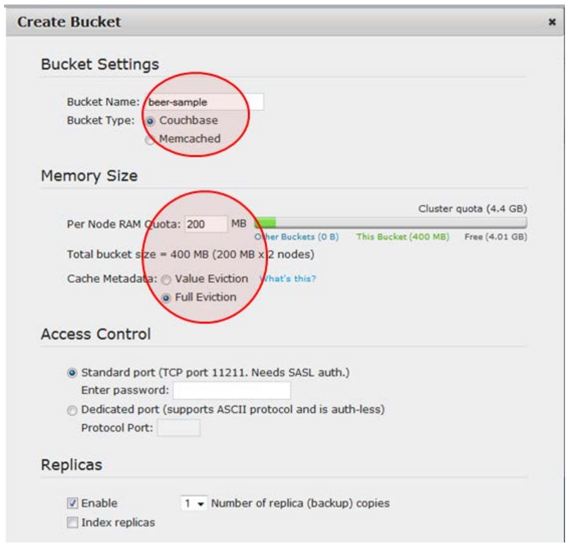
You should already be logged into the remote 2-node cluster (using XDCR Node #1's public hostname). Since we previously deleted the default bucket, there should be zero buckets configured. Click on Data Buckets at the top, then Create New Data Bucket:



No buckets currently defined! Click "Create New Data Bucket" above to create one.

On the Create Bucket popup, enter the following settings (only items in red need to be changed):

- **Bucket Name: beer-sample**
- **Bucket Type: Couchbase**
- **Per Node Ram Quota: 200 MB**
- **Cache Metadata Full Ejection**
- **Access Control: Standard port (do not enter a password!)**
- **Replicas: Enable and set to 1**
- **View Index Replicas: Unchecked**
- **Disk I/O Optimization High**
- **Auto-Compaction: leave unchecked**
- **Flush: Place a check to set it as Enabled**
- **Click Create**



Create Bucket ×

Bucket Settings

Bucket Name: beer-sample
Bucket Type: [✓ Couchbase] Memcached

Memory Size

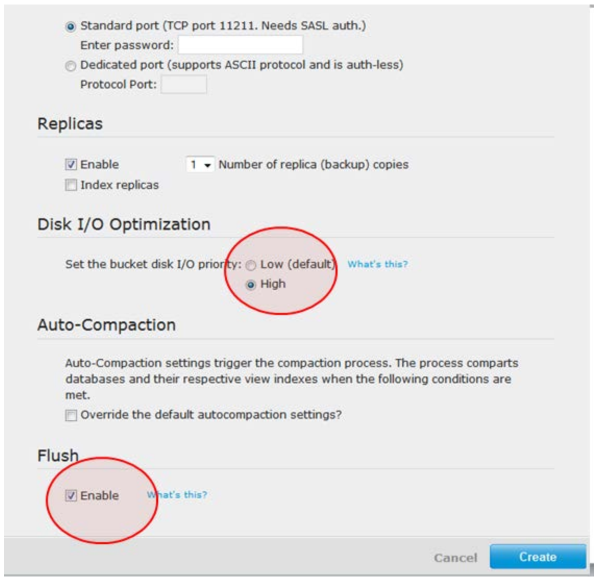
Cluster quota (4.4 GB)
Per Node RAM Quota: 200 MB
Other Buckets (0 B) This Bucket (400 MB) Free (4.01 GB)
Total bucket size = 400 MB (200 MB x 2 nodes)
Cache Metadata: Value Eviction [✓ Full Eviction] What's this?

Access Control

[✓ Standard port (TCP port 11211. Needs SASL auth.)]
Enter password:
Dedicated port (supports ASCII protocol and is auth-less)
Protocol Port:

Replicas

[✓ Enable] 1 Number of replica (backup) copies
Index replicas



[✓ Standard port (TCP port 11211. Needs SASL auth.)]
Enter password:
Dedicated port (supports ASCII protocol and is auth-less)
Protocol Port:

Replicas

[✓ Enable] 1 Number of replica (backup) copies
Index replicas

Disk I/O Optimization

Set the bucket disk I/O priority: Low (default) [✓ High] What's this?

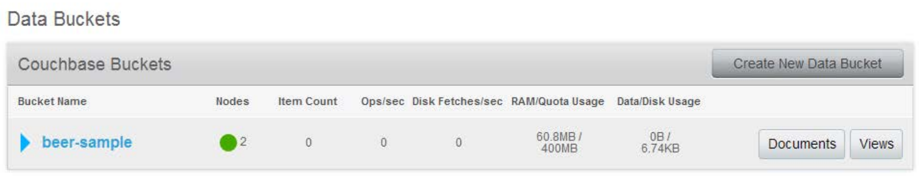
Auto-Compaction

Auto-Compaction settings trigger the compaction process. The process comparts databases and their respective view indexes when the following conditions are met.
Override the default autocompaction settings?

Flush

[✓ Enable] What's this?

You should now see the new, empty beer-sample bucket. Note that the Item count below is 0:



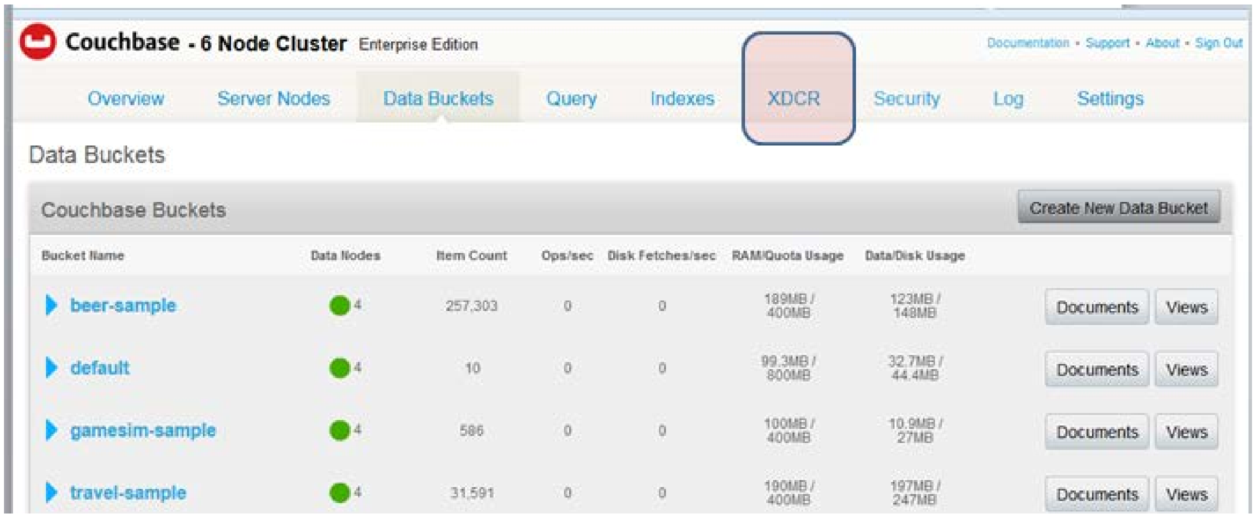
Data Buckets

Couchbase Buckets

Bucket Name	Nodes	Item Count	Ops/sec	Disk Fetches/sec	RAM/Quota Usage	Data/Disk Usage		
beer-sample	● 2	0	0	0	60.8MB / 400MB	0B / 6.74KB	Documents	Views

Now, switch to the Web UI for the production 6-node cluster. At this point, it will be useful to open 2 tabs in your browser window: one for the 6-node cluster and one for the 2-node cluster. For both clusters, connect to their Node #1's public hostname for each cluster.

Click on the XDCR button at the top of the screen in the production 6-node cluster:



Data Buckets

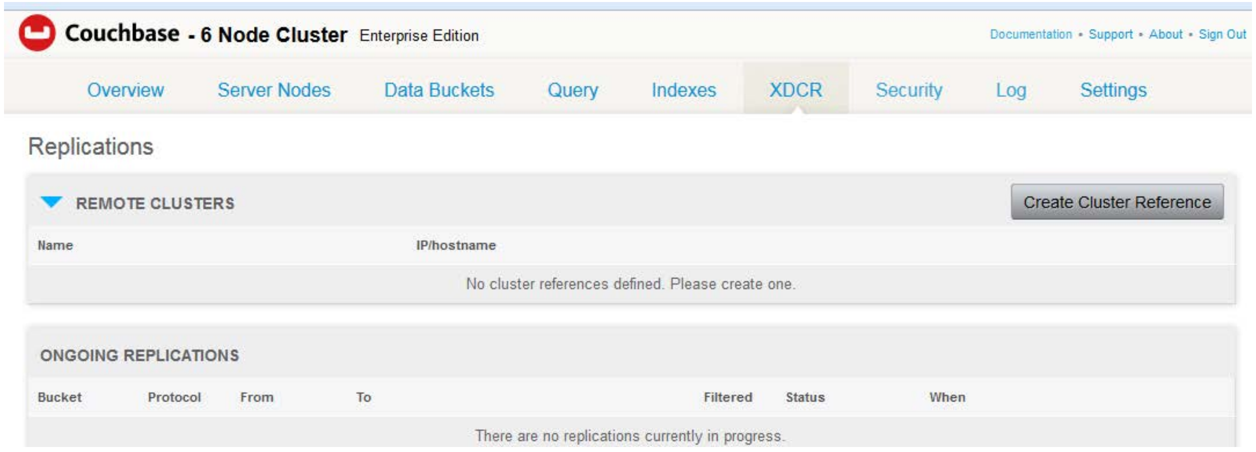
Couchbase Buckets

Bucket Name	Data Nodes	Item Count	Ops/sec	Disk Fetches/sec	RAM/Quota Usage	Data/Disk Usage
beer-sample	4	257,303	0	0	189MB / 400MB	123MB / 148MB
default	4	10	0	0	99.3MB / 800MB	32.7MB / 44.4MB
gamesim-sample	4	586	0	0	100MB / 400MB	10.9MB / 27MB
travel-sample	4	31,591	0	0	190MB / 400MB	197MB / 247MB

We will use the 6-node production cluster as the source cluster and the remote 2-node cluster as the destination.

To set up a destination cluster reference, click the Create Cluster Reference button:

Warning: you should be completing the next steps from the 6-node production cluster's Web UI (I recommend connecting to the web UI of the 1st node specifically)!



Replications

REMOTE CLUSTERS

Name	IP/hostname

No cluster references defined. Please create one.

ONGOING REPLICATIONS

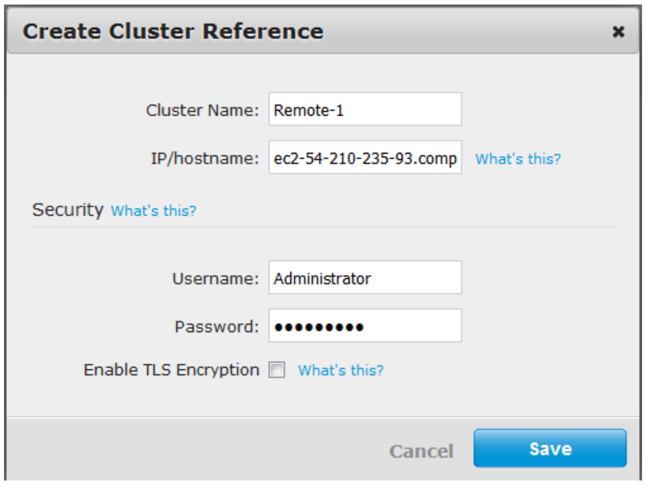
Bucket	Protocol	From	To	Filtered	Status	When

There are no replications currently in progress.

On the pop-up enter:

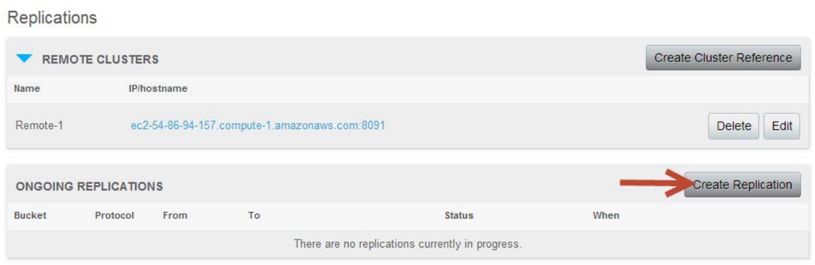
- **Cluster Name: Remote-1**
- **IP/hostname: <public hostname of the Node #1 (red) in the remote cluster>**
- **Username: Administrator**
- **Password: couchbase**
- **Enable Encryption: unchecked**

Click Save



Cluster Name: Remote-1
IP/hostname: ec2-54-210-235-93.comp
Username: Administrator
Password: •••••••••
Enable TLS Encryption ☐

The Web UI will now show a remote cluster configured as Remote-1. Click on Create Replication to set up the unidirectional replication for the beer-sample bucket:



Replications

REMOTE CLUSTERS

Name	IP/hostname		
Remote-1	ec2-54-86-94-157.compute-1.amazonaws.com:8091	Delete	Edit

ONGOING REPLICATIONS

Bucket	Protocol	From	To	Status	When

There are no replications currently in progress.

On the Create Replication pop-up, edit the following three settings:

Replicate changes from:

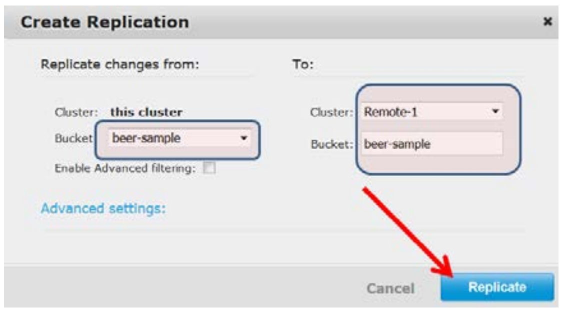
Bucket: beer-sample

To:

Cluster: Remote-1

Bucket: beer-sample

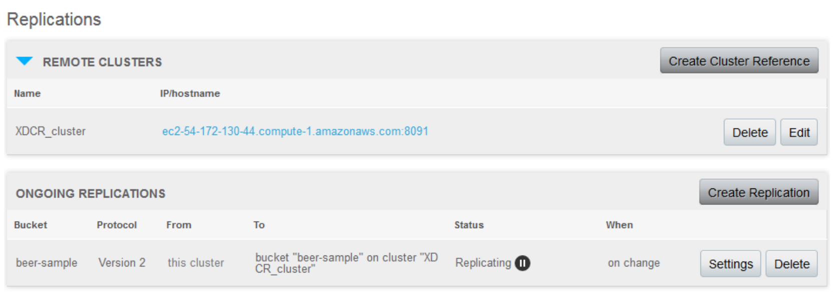
Click Replicate:



Replicate changes from:
Cluster: this cluster
Bucket: beer-sample
Enable Advanced filtering: ☐

To:
Cluster: Remote-1
Bucket: beer-sample

The Web UI will now update to show that there is an on-going replication occurring using Protocol version 2 for beer-sample from this cluster to a remote cluster Remote-1. You should see the Status as "Replicating"
Notice new pause button next to Replicating



Replications

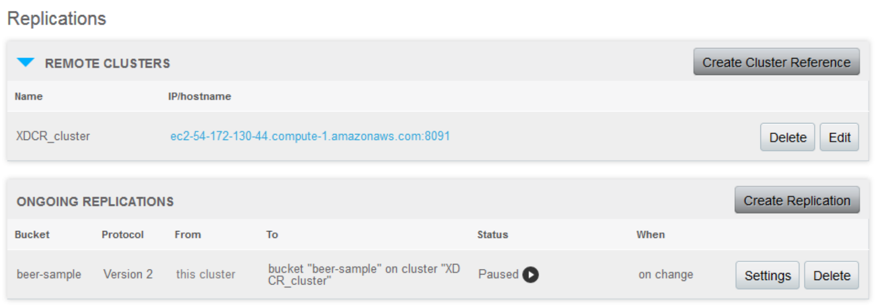
REMOTE CLUSTERS — Create Cluster Reference

Name	IP/hostname	
XDCR_cluster	ec2-54-172-130-44.compute-1.amazonaws.com:8091	Delete Edit

ONGOING REPLICATIONS — Create Replication

Bucket	Protocol	From	To	Status	When	
beer-sample	Version 2	this cluster	bucket "beer-sample" on cluster "XDCR_cluster"	Replicating ⏸	on change	Settings Delete

Replication can be paused as shown in this screen capture.



Replications

REMOTE CLUSTERS — Create Cluster Reference

Name	IP/hostname	
XDCR_cluster	ec2-54-172-130-44.compute-1.amazonaws.com:8091	Delete Edit

ONGOING REPLICATIONS — Create Replication

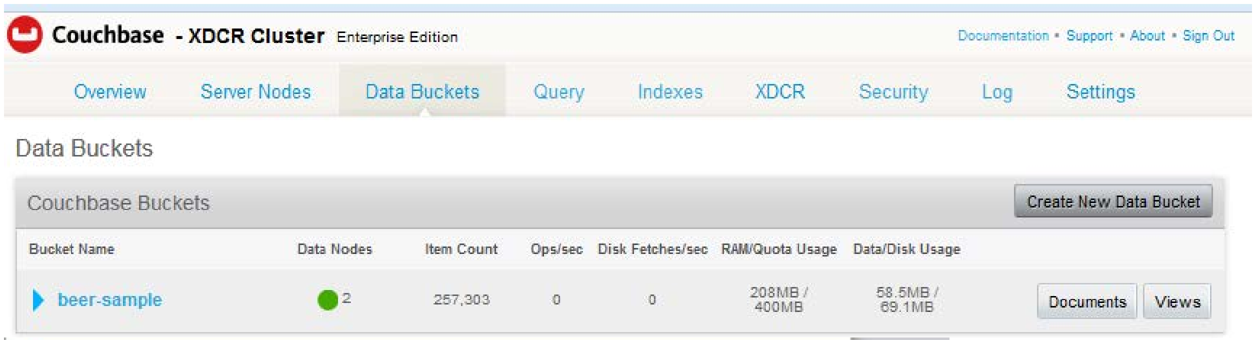
Bucket	Protocol	From	To	Status	When	
beer-sample	Version 2	this cluster	bucket "beer-sample" on cluster "XDCR_cluster"	Paused ▶	on change	Settings Delete

Note that the "When" column setting is currently blank for When with Couchbase 4.5.0. So there's no way to change this currently. In a future version of Couchbase there may be different options to choose from.

Let's check if the replication as really occurred. Switch to the 2nd tab in your browser, which should be for Node #1 (red) in the remote 2-node cluster. Click Refresh in the browser and under the Data Buckets screen, you should see the same number of items under beer-sample as in 6 node cluster.

(Note that if you verify this same # in the production 6-node cluster, you will also see the same value. Screen caps show 257,303,yours may vary.

Click on the Documents button next to the beer-sample bucket:

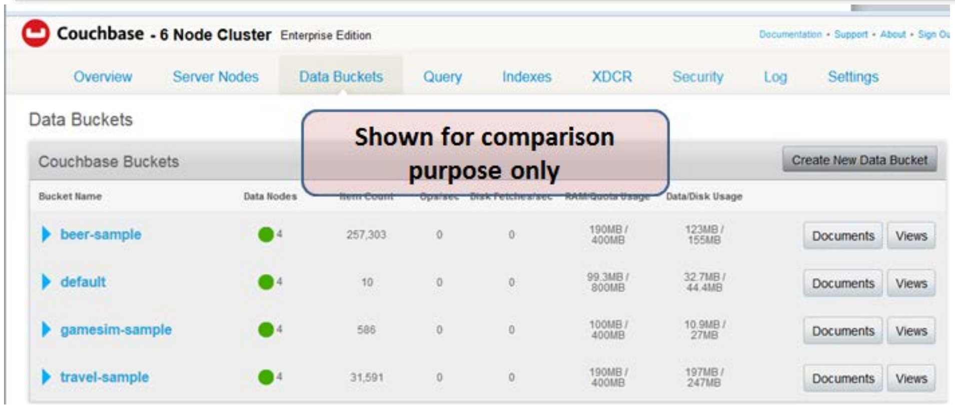


Couchbase - XDCR Cluster Enterprise Edition

Data Buckets

Couchbase Buckets

Bucket Name	Data Nodes	Item Count	Ops/sec	Disk Fetches/sec	RAM/Quota Usage	Data/Disk Usage
beer-sample	● 2	257,303	0	0	208MB / 400MB	58.5MB / 69.1MB



Couchbase - 6 Node Cluster Enterprise Edition

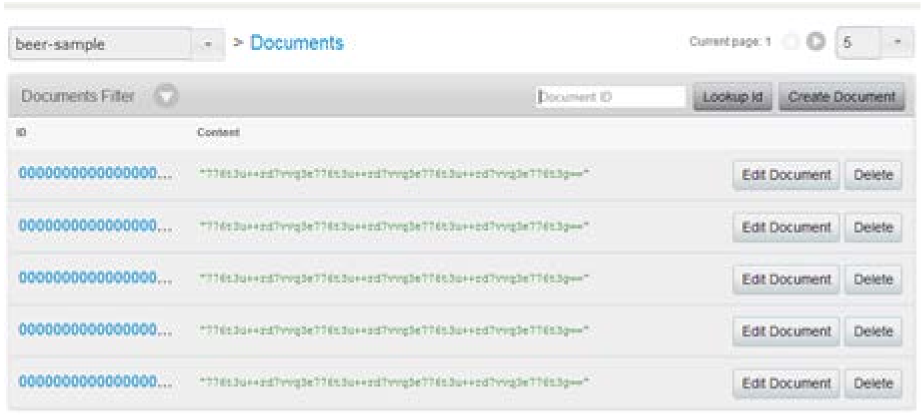
Data Buckets

Shown for comparison purpose only

Couchbase Buckets

Bucket Name	Data Nodes	Item Count	Ops/sec	Disk Fetches/sec	RAM/Quota Usage	Data/Disk Usage
beer-sample	● 4	257,303	0	0	190MB / 400MB	123MB / 155MB
default	● 4	10	0	0	99.3MB / 800MB	32.7MB / 44.4MB
gamesim-sample	● 4	586	0	0	100MB / 400MB	10.9MB / 27MB
travel-sample	● 4	31,591	0	0	190MB / 400MB	197MB / 247MB

Under Documents, you may see a lot of random data, not pertaining to beer-sample JSON files!
This is because in the previous lab, we used cbworkloadgen to push 250,000 writes into this bucket to observe the indexing of the views.

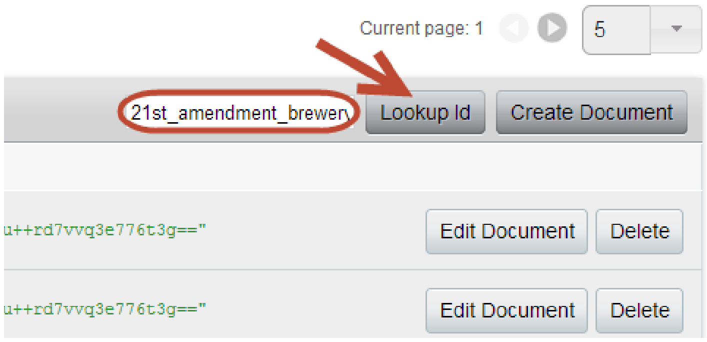


beer-sample > Documents

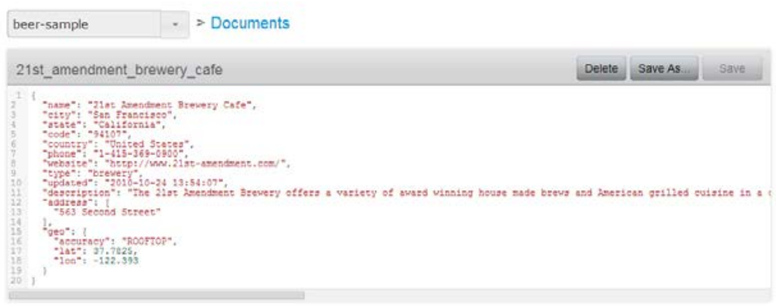
Current page: 1

ID	Content
0000000000000000...	"776t3u++zd7vvg3e776t3u++zd7vvg3e776t3u++zd7vvg3e776t3g=="
0000000000000000...	"776t3u++zd7vvg3e776t3u++zd7vvg3e776t3u++zd7vvg3e776t3g=="
0000000000000000...	"776t3u++zd7vvg3e776t3u++zd7vvg3e776t3u++zd7vvg3e776t3g=="
0000000000000000...	"776t3u++zd7vvg3e776t3u++zd7vvg3e776t3u++zd7vvg3e776t3g=="
0000000000000000...	"776t3u++zd7vvg3e776t3u++zd7vvg3e776t3u++zd7vvg3e776t3g=="

You can verify that the original JSON data that came with the beer-sample bucket is still in this bucket by searching for a specific brewery. In the search box for this bucket, enter the document ID '`21st_amendment_brewery_cafe`' (without the single quote marks) and click on Lookup id:

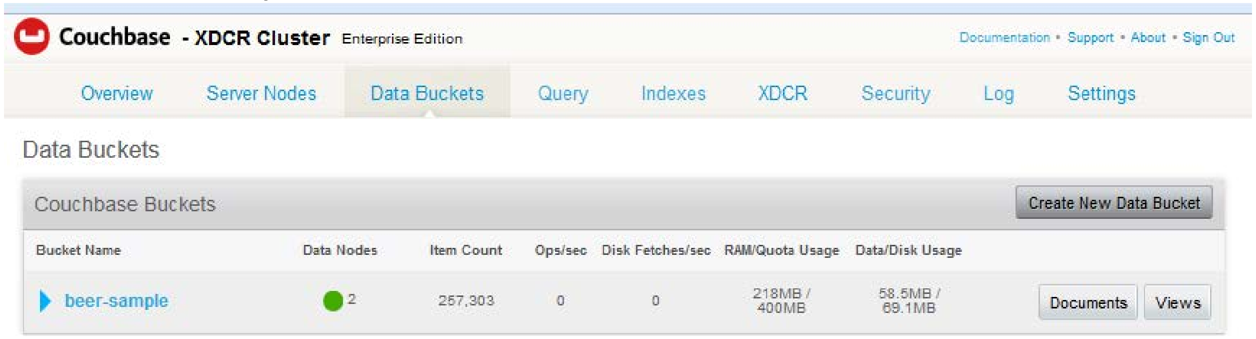


The specific brewery that you requested will now be displayed:



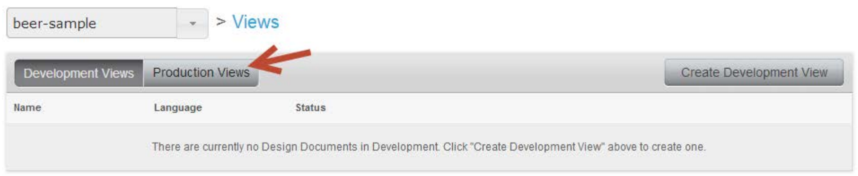
```
{
  "name": "21st Amendment Brewery Cafe",
  "city": "San Francisco",
  "state": "California",
  "code": "94107",
  "country": "United States",
  "phone": "1-415-369-0900",
  "website": "http://www.21st-amendment.com/",
  "type": "brewery",
  "updated": "2010-10-24 13:54:07",
  "description": "The 21st Amendment Brewery offers a variety of award winning house made brews and American grilled cuisine in a
  "address": [
    "563 Second Street"
  ],
  "geo": {
    "accuracy": "ROOFTOP",
    "lat": 37.7825,
    "lon": -122.393
  }
}
```

Remain within the Web UI of the remote cluster and click on Data Buckets and then click Views for beer-sample:



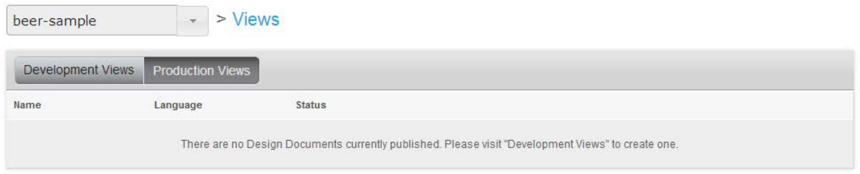
Bucket Name	Data Nodes	Item Count	Ops/sec	Disk Fetches/sec	RAM/Quota Usage	Data/Disk Usage
beer-sample	● 2	257,303	0	0	218MB / 400MB	58.5MB / 69.1MB

Notice that there are no Development Views on the remote cluster for beer-sample. Click on Production Views:



There are currently no Design Documents in Development. Click "Create Development View" above to create one.

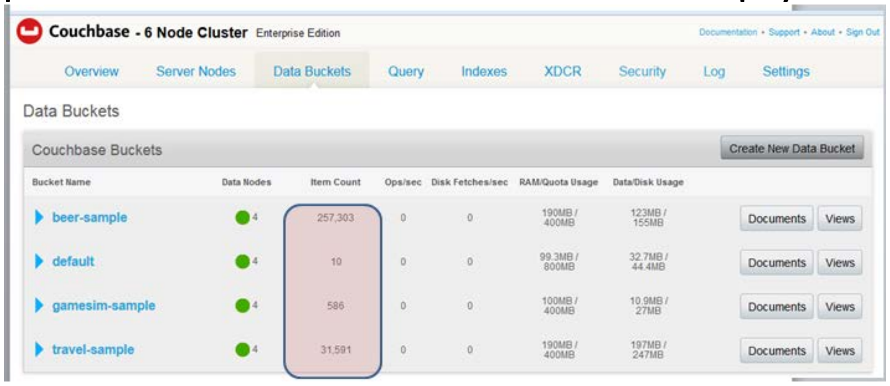
Once again, notice that there are no Production Views for beer-sample on the remote-cluster:



Name	Language	Status
There are no Design Documents currently published. Please visit "Development Views" to create one.		

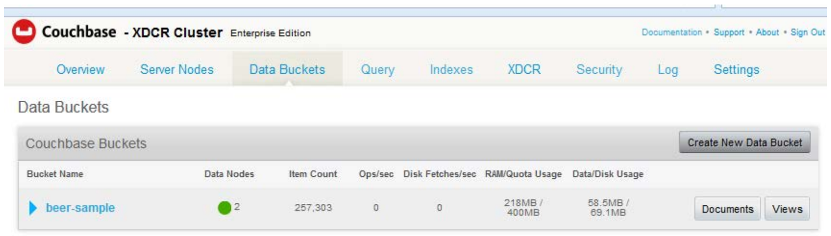
The Design Documents (and their Views) cannot be replicated automatically or using the Web UI. Instead you have to use the cbbackup and cbrestore tools to handle the Design Documents or you have to manually re-create the Design Document (with its Views) on the remote cluster. We will try the cbbackup/cbrestore tools in the next Lab #7. So, hold this thought and continue with the remainder of this lab.

At this point in the lab you should have 3-4 buckets(some may have travel-sample) on the production 6-node cluster side with the item counts displayed below:



Bucket Name	Data Nodes	Item Count	Ops/sec	Disk Fetches/sec	RAM/Quota Usage	Data/Disk Usage	
beer-sample	● 4	257,303	0	0	190MB / 400MB	123MB / 155MB	Documents Views
default	● 4	10	0	0	99.3MB / 800MB	32.7MB / 44.4MB	Documents Views
gamesim-sample	● 4	586	0	0	100MB / 400MB	10.9MB / 27MB	Documents Views
travel-sample	● 4	31,591	0	0	190MB / 400MB	197MB / 247MB	Documents Views

And the remote 2-node cluster should have just 1 bucket (which we're replicating to) with the same item count as its production partner:



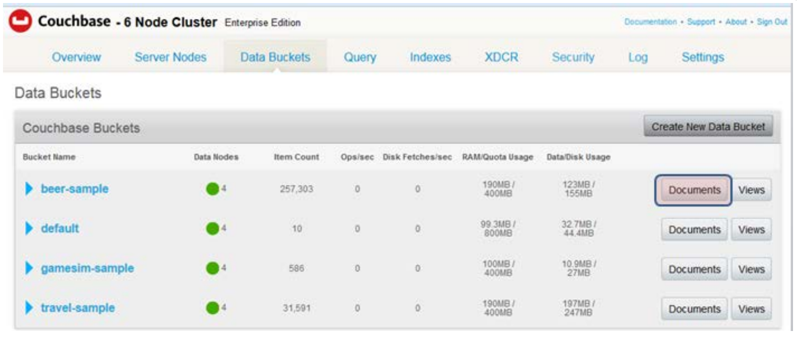
Bucket Name	Data Nodes	Item Count	Ops/sec	Disk Fetches/sec	RAM/Quota Usage	Data/Disk Usage	
beer-sample	● 2	257,303	0	0	218MB / 400MB	58.5MB / 69.1MB	Documents Views

Understanding which keys will be replicated with XDCR and exploring the revision count parameter:

Some of the algorithms that XDCR uses could be non-intuitive. So, in this section we will create items on both the source and destination cluster and check to see if it gets replicated. We will also learn about how the revision count is used to decide which items to push from source -> destination.

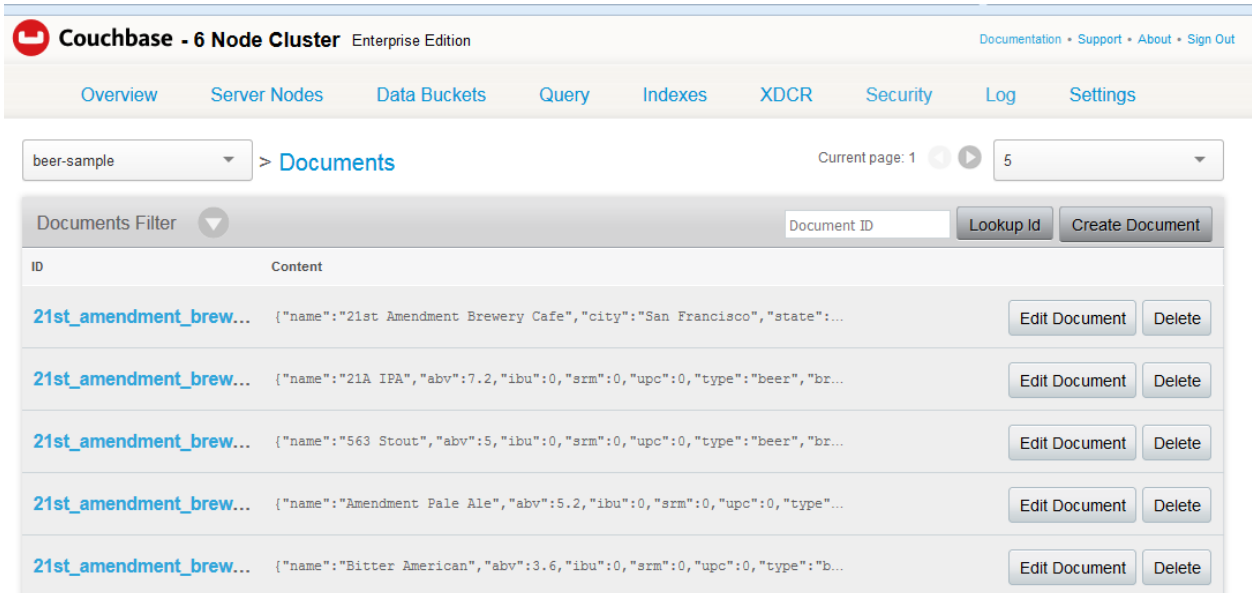
First, let's create a new key/document on the source side under beer-sample and then check to see if this fresh item gets replicated to the destination cluster.

On the source (6-node production cluster's browser tab), click on Data Buckets at the top and then click on the 'Documents' button next for the beer-sample bucket:



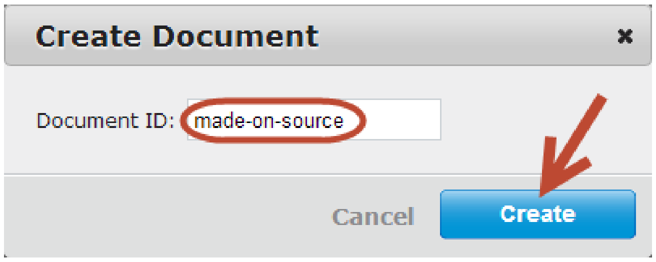
Bucket Name	Data Nodes	Item Count	Ops/sec	Disk Fetches/sec	RAM/Quota Usage	Data/Disk Usage		
beer-sample	● 4	257,303	0	0	190MB / 400MB	123MB / 155MB	Documents	Views
default	● 4	10	0	0	99.3MB / 800MB	32.7MB / 44.4MB	Documents	Views
gamesim-sample	● 4	586	0	0	100MB / 400MB	10.9MB / 27MB	Documents	Views
travel-sample	● 4	31,591	0	0	190MB / 400MB	197MB / 247MB	Documents	Views

Click Create Document:



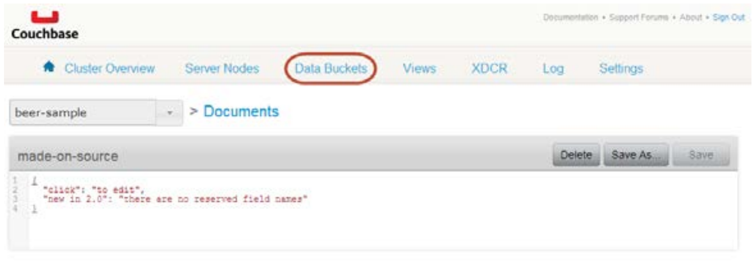
ID	Content		
21st_amendment_brew...	{"name":"21st Amendment Brewery Cafe","city":"San Francisco","state":...	Edit Document	Delete
21st_amendment_brew...	{"name":"21A IPA","abv":7.2,"ibu":0,"srm":0,"upc":0,"type":"beer","br...	Edit Document	Delete
21st_amendment_brew...	{"name":"563 Stout","abv":5,"ibu":0,"srm":0,"upc":0,"type":"beer","br...	Edit Document	Delete
21st_amendment_brew...	{"name":"Amendment Pale Ale","abv":5.2,"ibu":0,"srm":0,"upc":0,"type"...	Edit Document	Delete
21st_amendment_brew...	{"name":"Bitter American","abv":3.6,"ibu":0,"srm":0,"upc":0,"type":"b...	Edit Document	Delete

In the pop-up, name the document 'made-on-source' and click "Create":



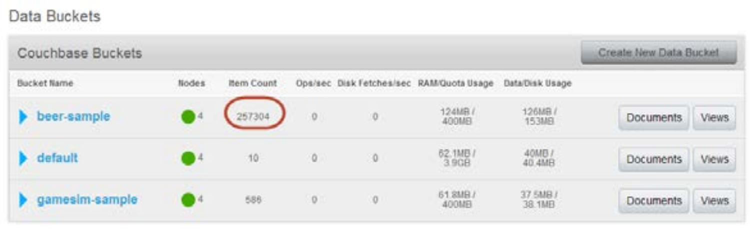
Document ID: made-on-source

The new document will be created in the beer-sample bucket. Do not edit and resave this document! Just leave the JSON the way it is in its default state and click on Data Buckets at the top of the screen:



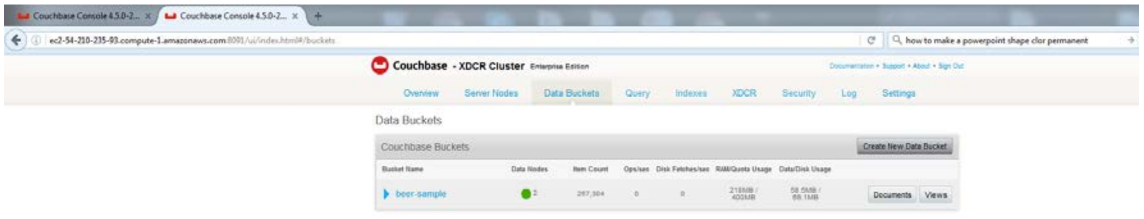
```
{
  "click": "to edit",
  "new in 2.0": "there are no reserved field names"
}
```

You should see the item count field for the beer-sample bucket as increased by one to 257,304:



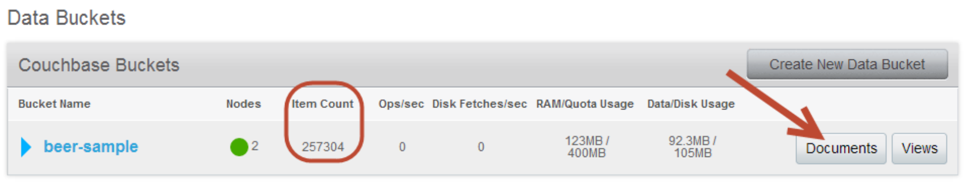
Bucket Name	Nodes	Item Count	Ops/sec	Disk Fetches/sec	RAM/Quota Usage	Data/Disk Usage		
beer-sample	● 4	257304	0	0	124MB / 400MB	126MB / 153MB	Documents	Views
default	● 4	10	0	0	62.1MB / 3.9GB	40MB / 40.4MB	Documents	Views
gamesim-sample	● 4	586	0	0	61.8MB / 400MB	37.5MB / 38.1MB	Documents	Views

Next, switch to the browser tab for the remote/destination 2-node cluster and refresh the page(on the Data Buckets tab):


Couchbase Console 4.5.0-2...
Couchbase Console 4.5.0-2...
ec2-54-210-235-93.compute-1.amazonaws.com:8091/ui/index.html#/buckets
how to make a powerpoint shape clor permanent
Couchbase - XDCR Cluster Enterprise Edition
Documentation • Support • About • Sign Out
Overview
Server Nodes
Data Buckets
Query
Indexes
XDCR
Security
Log
Settings
Data Buckets
Couchbase Buckets
Create New Data Bucket

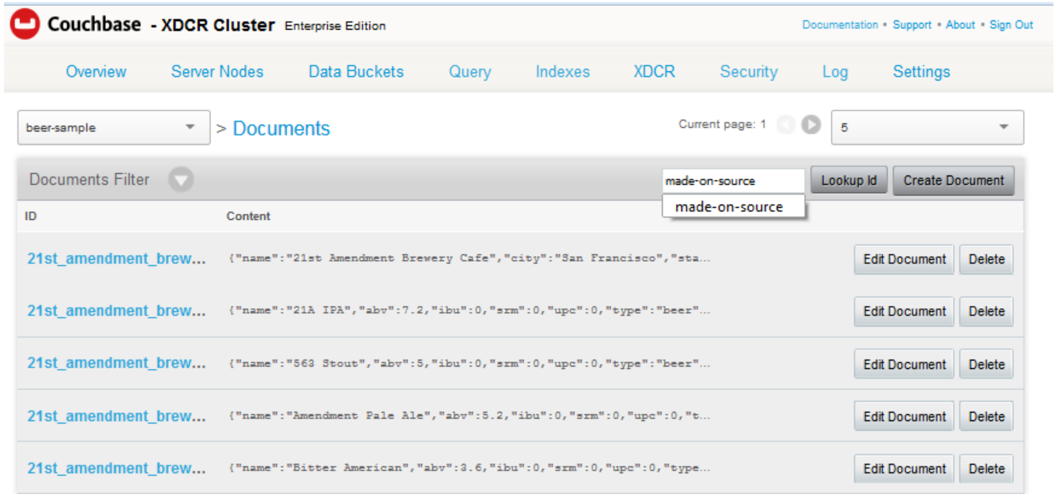
Bucket Name	Data Nodes	Item Count	Ops/sec	Disk Fetches/sec	RAM/Quota Usage	Data/Disk Usage		
beer-sample	● 2	257,304	0	0	218MB / 400MB	58.5MB / 68.1MB	Documents	Views

You will see the item count will refresh on the remote side to 257,304 items, matching the source side. Click on Documents so we can attempt reading the new document:



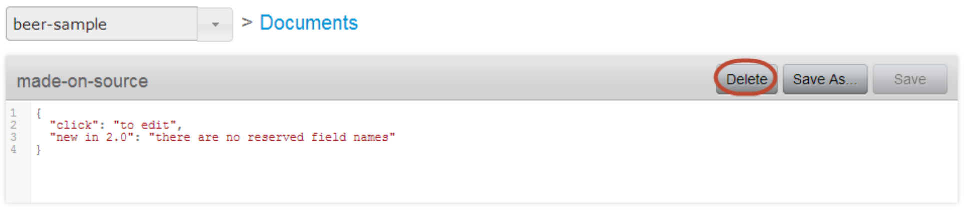
Bucket Name	Nodes	Item Count	Ops/sec	Disk Fetches/sec	RAM/Quota Usage	Data/Disk Usage		
beer-sample	● 2	257304	0	0	123MB / 400MB	92.3MB / 105MB	Documents	Views

In the search field, type 'made-on-source' (without the quotes) and click "Lookup id":



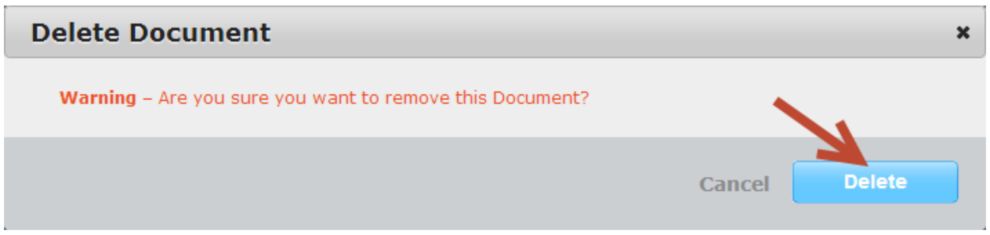
ID	Content
21st_amendment_brew...	{"name":"21st Amendment Brewery Cafe","city":"San Francisco","sta...
21st_amendment_brew...	{"name":"21A IPA","abv":7.2,"ibu":0,"srm":0,"upc":0,"type":"beer"...
21st_amendment_brew...	{"name":"563 Stout","abv":5,"ibu":0,"srm":0,"upc":0,"type":"beer"...
21st_amendment_brew...	{"name":"Amendment Pale Ale","abv":5.2,"ibu":0,"srm":0,"upc":0,"t...
21st_amendment_brew...	{"name":"Bitter American","abv":3.6,"ibu":0,"srm":0,"upc":0,"type...

You should now see the document you created on the source side. Delete this document (destination side/2 node cluster)by clicking Delete and then let's see if it reappears on the destination side:



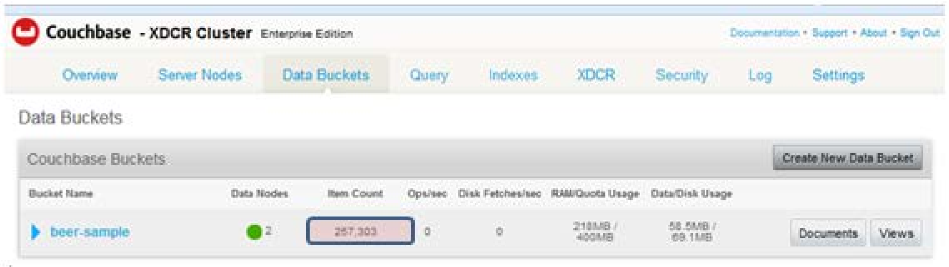
```
{
  "click": "to edit",
  "new in 2.0": "there are no reserved field names"
}
```

Click 'Delete' on the pop-up to confirm the deletion:



Warning – Are you sure you want to remove this Document?

Finally, click on Data Buckets at the top and verify that the item count as been reduced by 1 to 257,303 (*you may have to refresh the browser page*):



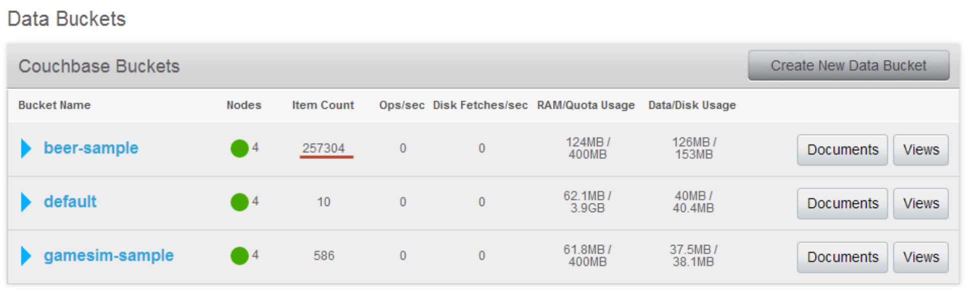
Couchbase - XDCR Cluster Enterprise Edition

Data Buckets

Couchbase Buckets

Bucket Name	Data Nodes	Item Count	Ops/sec	Disk Fetches/sec	RAM/Quota Usage	Data/Disk Usage
beer-sample	● 2	257,303	0	0	218MB / 400MB	58.5MB / 69.1MB

Switch browser tabs to the production/source 6-node cluster and refresh the "Data Buckets" page to verify that there are still 257,304 items on this side:



Data Buckets

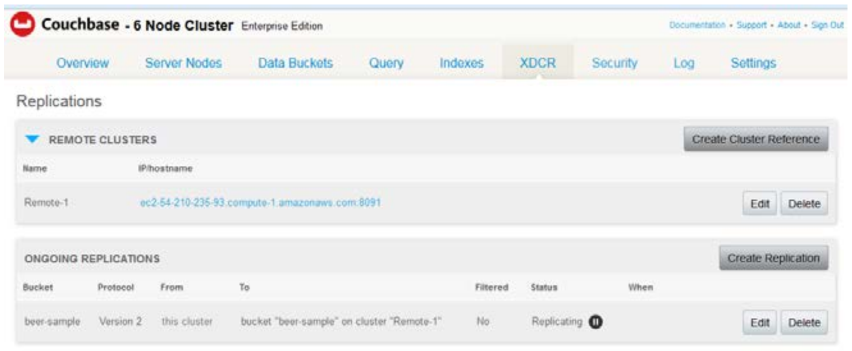
Couchbase Buckets

Bucket Name	Nodes	Item Count	Ops/sec	Disk Fetches/sec	RAM/Quota Usage	Data/Disk Usage
beer-sample	● 4	257304	0	0	124MB / 400MB	126MB / 153MB
default	● 4	10	0	0	62.1MB / 3.9GB	40MB / 40.4MB
gamesim-sample	● 4	586	0	0	61.8MB / 400MB	37.5MB / 38.1MB

So at this point, even if you wait 5 minutes the item that we deleted on the destination cluster will NOT get automatically replicated back to the destination. Why is this?

First, let's go and look at the advanced settings in the Web UI for the ongoing replication to see if there's something we can change to fix this issue.

On the source/production 6-node cluster, click on XDCR at the top of the page and then click "Edit" for the beer-sample replication:



Couchbase - 6 Node Cluster Enterprise Edition

Replications

REMOTE CLUSTERS

Name	IP/hostname
Remote-1	ec2-54-210-235-93.compute-1.amazonaws.com:8091

ONGOING REPLICATIONS

Bucket	Protocol	From	To	Filtered	Status	When
beer-sample	Version 2	this cluster	bucket "beer-sample" on cluster "Remote-1"	No	Replicating	

Notice above that this replication is using Version 2 of the protocol.

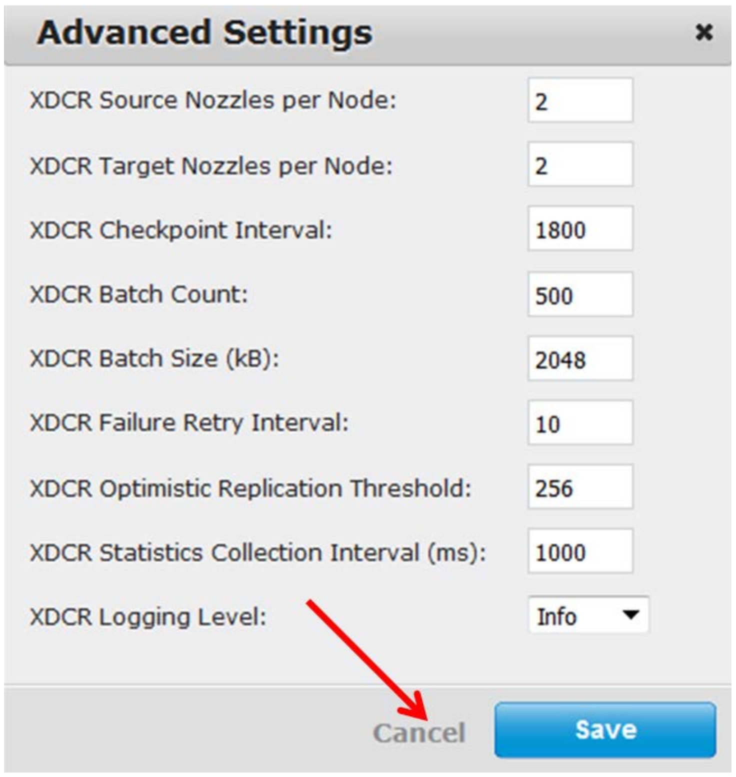
The XDCR protocol defaults to version 2.

- Version 1 uses the REST protocol for replication. This increases XDCR throughput at destination clusters. If you use the Elasticsearch plug-in, which depends on XDCR, choose version 1.

- Version 2 uses memcached REST protocol for replication. It is a high-performance mode that directly uses the memcached protocol on destination nodes. Choose version 2 when setting up a new replication with Couchbase Server 2.2 or later.

You can change this setting via the REST API for XDCR internal settings or the couchbase-cli tool. Alternatively, when you originally create the replication, you can expand the advanced settings and choose version 1 or 2. In most cases, you will go with Version 2.

The Advanced Settings pop-up will be displayed. Here is a quick explanation of these settings. After reading through the explanations below, click "Cancel":


Advanced Settings

XDCR Source Nozzles per Node:	2
XDCR Target Nozzles per Node:	2
XDCR Checkpoint Interval:	1800
XDCR Batch Count:	500
XDCR Batch Size (kB):	2048
XDCR Failure Retry Interval:	10
XDCR Optimistic Replication Threshold:	256
XDCR Statistics Collection Interval (ms):	1000
XDCR Logging Level:	Info

XDCR Source and Target Nozzles per node.

XDCR Checkpoint Interval:

Interval between checkpoints, 60 to 14400 (seconds). Default 1800. At this time interval, batches of data via XDCR replication will be placed in the front of the disk persistence queue. Changing this to a smaller value could impact cluster operations when you have significant amount of write operations on a destination cluster and you are performing bidirectional replication with XDCR. For instance, if you set this to 5 minutes, the incoming batches of data via XDCR replication will take priority in the disk write queue over incoming write workload for a destination cluster. This may result in the problem of having an ever growing disk-write queue on a destination cluster; also items in the disk-write queue that are higher priority than the XDCR items will grow staler/older before they are persisted.

XDCR Batch Count:

Document batching count, 500 to 10000. Default 500. In general, increasing this value by 2 or 3 times will improve XDCR transmissions rates, since larger batches of data will be sent in the same timed interval. For unidirectional replication from a source to a destination cluster, adjusting this setting by 2 or 3 times will improve overall replication performance as long as persistence to disk is fast enough on the destination cluster.

XDCR Batch Size (KB):

Document batching size, 10 to 100000 (KB). Default 2048. In general, increasing this value by 2 or 3 times will improve XDCR transmissions rates, since larger batches of data will be sent in the same timed interval.

XDCR Failure Retry Interval:

Interval for restarting failed XDCR, 1 to 300 (seconds). Default 30. If you expect more frequent network or server failures, you may want to set this to a lower value. This is the time that XDCR waits before it attempts to restart replication after a server or network failure.

XDCR Optimistic Replication Threshold:

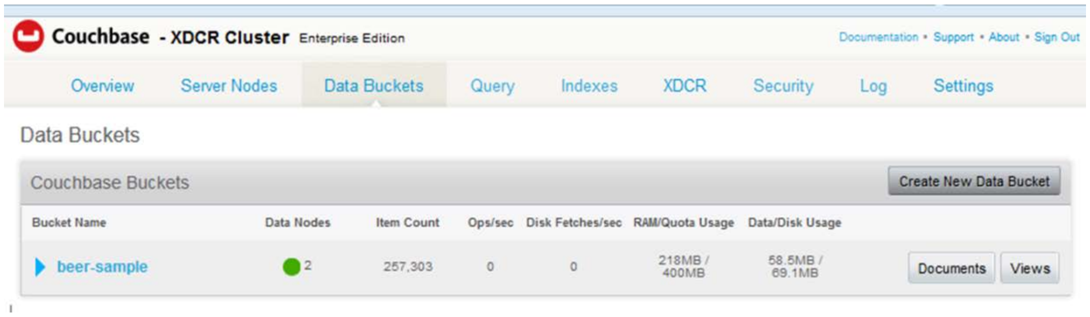
This will improve latency for XDCR.

This is document size in bytes. 0 to 2097152 Bytes (20MB). Default is 256 Bytes. XDCR will get metadata for documents larger than this size on a single time before replicating the document to a destination cluster.

- - - -

None of the above settings really helps explain why the document named with the key 'made-on-source' is not being automatically re-replicated from source -> destination since we deleted it on the destination.

It has probably been about 3 minutes since you deleted the item at <Destination>. **If you switch over to the remote/destination cluster's Web UI tab and refresh the 'Data Buckets' page, you will still not see this item:**



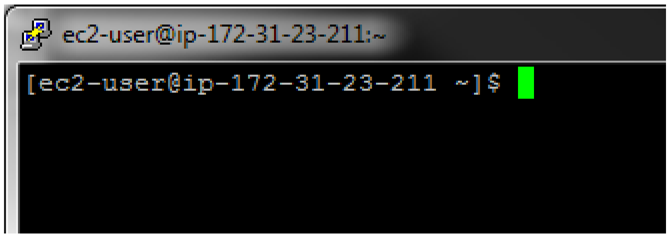
Bucket Name	Data Nodes	Item Count	Ops/sec	Disk Fetches/sec	RAM/Quota Usage	Data/Disk Usage
beer-sample	● 2	257,303	0	0	218MB / 400MB	58.5MB / 69.1MB

How do we explain this issue? Why is that document with the key 'made-on-source' not getting re-replicated?

To explain this, we need to do some investigating. Let's check what the revision # is for the item with the key 'made-on-source' in the source cluster and in the destination cluster. Our hunch should currently be that when we deleted the item in the destination cluster, we actually just updated the document with a tombstone as the value and incremented its revision count # to 2.
The revision count # for the original document created on the source side is probably still set to 1. So the source cluster will simply not push the item to the destination side b/c the revision # is higher on the destination.

But let's check this logic via an investigation.

Connect to the App Server (black VM):



```
[ec2-user@ip-172-31-23-211 ~]$
```

Run the 'cbc cat' command to first simply read the item from the source cluster. Provide the command the public hostname of Node #1 (dark blue VM) in the 6-node production/source cluster:

```
[ec2-user@ip-172-31-23-211 ~]$ cbc-cat -U couchbase://ec2-54-172-130-66.compute-1.amazonaws.com/beer-sample made-on-source
made-on-source       CAS=0x679438856414, Flags=0x0. Size=67
{"click":"to edit","with JSON":"there are no reserved field names"}
```

Then run the 'cbc hash' command to find out where this key is actually stored in this 6-node cluster (again provide the command the public hostname of Node #1 on the production side):

```
[ec2-user@ip-172-31-23-211 ~]$ cbc-hash -U couchbase://ec2-54-172-130-
66.compute-1.amazonaws.com/beer-sample made-on-source
made-on-source: [vBucket=788, Index=3] Server: ec2-54-209-38-
55.compute-1.amazonaws.com:11210, CouchAPI: http://ec2-54-209-38-
55.compute-1.amazonaws.com:8092/beer-sample
Replica #0: Index=0, Host=ec2-54-152-187-112.compute-
1.amazonaws.com:11210
```

You can see this in my specific setup, the active copy of the key is hashed to vBucket # 788 on the Server with the public hostname "ec2-54-209-38-55.compute-1.amazonaws.com". In my setup, this translates to the 3rd Node (Green VM) in the 6-node cluster.

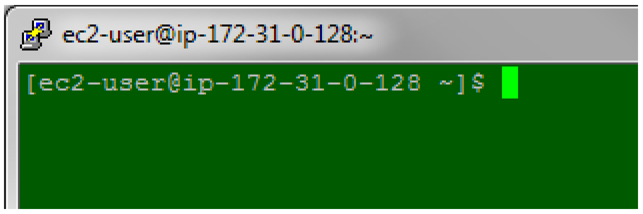
Write down the specific vBucket #, hostname and shell window color for your environment:

vBucket #: ____________________
Public Hostname: _____________________
Hostname shell color: _________________

Since in my specific setup this key hashed to the 3rd/Green node, I will go to the green window for the next step.

Warning: Even though I'm switching to the Green node, you may have to go to the dark blue, light blue, green or yellow VM in your specific environment!

Switch to the node where the 'made-on-source' key's active replica hashed to:



```
ec2-user@ip-172-31-0-128:~
[ec2-user@ip-172-31-0-128 ~]$
```

Print a listing of all the data files for the beer-sample bucket on this node:

```
[ec2-user@ip-172-31-0-128 ~]$ sudo ls -al
/opt/couchbase/var/lib/couchbase/data/beer-sample
total 43744
drwxrwx---. 2 couchbase couchbase   20480 Jun  2 15:40 .
drwxrwx---. 7 couchbase couchbase    4096 May 31 13:25 ..
-rw-rw----. 1 couchbase couchbase   53339 May 31 17:41 1000.couch.2
-rw-rw----. 1 couchbase couchbase   49243 May 31 17:41 1001.couch.2
-rw-rw----. 1 couchbase couchbase   49243 May 31 17:41 1002.couch.2
-rw-rw----. 1 couchbase couchbase   49243 May 31 17:41 1003.couch.2
-rw-rw----. 1 couchbase couchbase   57435 May 31 17:41 1004.couch.2
-rw-rw----. 1 couchbase couchbase   49243 May 31 17:41 1005.couch.2
-rw-rw----. 1 couchbase couchbase   45147 May 31 17:41 1006.couch.2
<output truncated>
```

Well, that's a lot of results. (You'll probably see around 520 lines). I am specifically interested in vBucket 788 b/c that's where the key is supposed to be. Note, the vBucket # will be the same in your environment b/c the key will hash to the same vBucket #, however the vBucket might be hosted on a different node in your cluster.

Print a listing of all the files again, this time grepping for the specific vBucket # you are interested in:

```
[ec2-user@ip-172-31-0-128 ~]$ sudo ls -al
/opt/couchbase/var/lib/couchbase/data/beer-sample | grep 788
-rw-rw----. 1 couchbase couchbase   57435 Jun  2 14:54 788.couch.2
```

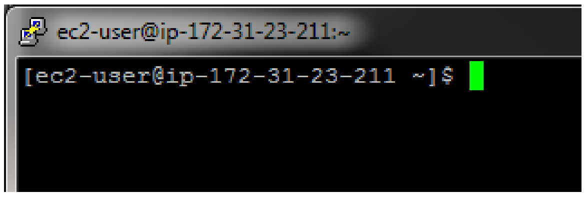
Note that in my environment the #2 at the end of the file name means that this vBucket file has gone through compaction once. The original vBucket file was named "788.couch.1".

Next let's take a look inside the file to look for the specific key 'made-on-source'. (Remember that you may have to issue a different file name and a different vBucket #!):

```
[ec2-user@ip-172-31-0-128 ~]$ sudo /opt/couchbase/bin/couch_dbdump
/opt/couchbase/var/lib/couchbase/data/beer-sample/788.couch.1 | grep -
B 2 -A 6 made-on-source
data: (snappy) VTKGNKUHMP
Doc seq: 245
     id: made-on-source
     rev: 1
     content_meta: 128
     size (on disk): 78
     cas: 1469445856484065280, expiry: 0, flags: 0, datatype: 1,
conflict_resolution_mode: 0
     size: 67
     data: (snappy) {"click":"to edit","with JSON":"there are no
reserved field names"}
```

Okay, great. So the revision count for the item in the source cluster is set to 1. This makes sense as I simply created the item and never updated or deleted it.
Let's try to find the revision count for the same item in the remote/destination cluster.

Switch to the App Server (black VM) to run the 'cbc hash' command:



```
[ec2-user@ip-172-31-23-211 ~]$
```

Run the 'cbc cat' command (on the destination cluster) to first simply read the item. Provide the command the public hostname of Node #1 (Red VM) in the 2-node remote/destination cluster:

```
[ec2-user@ip-172-31-23-211 ~]$ cbc-cat -U couchbase://ec2-54-86-94-
x.compute-1.amazonaws.com/beer-sample  made-on-source
Failed to get "made-on-source": The key does not exist on the server
```

Well, you can't read the key here b/c you deleted it earlier, but the tombstone for the key should still exist. Let's check.

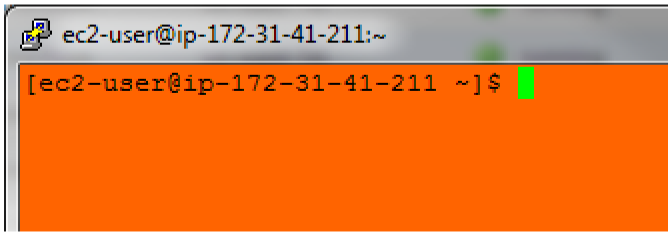
Run the 'cbc hash' command to find out where this key is actually stored in this 2-node cluster (again provide the command the public hostname of Node #1 on the remote/destination side):

```
[ec2-user@ip-172-31-23-211 ~]$ cbc-hash -U couchbase://ec2-54-86-94-
x.compute-1.amazonaws.com/beer-sample made-on-source
made-on-source: [vBucket=788, Index=0] Server: ec2-52-4-2-240.compute-
1.amazonaws.com:11210, CouchAPI: http://ec2-52-4-2-240.compute-
1.amazonaws.com:8092/beer-sample
Replica #0: Index=1, Host=ec2-52-7-140-134.compute-
1.amazonaws.com:11210
```

The vBucket ID is the same here, #788 for my specific environment. I now also know which server is supposed to be hosting the active copy of this item.

In my environment that host name (`ec2-54-86-96-38.compute-1.amazonaws.com`**) corresponds to Node #2 in the remote/destination side, which is the orange node.**

Although I am going to be switching to the Node #2's shell (Orange VM), you will have to either switch to the Red node (#1) or the Orange node (#2) in your environment:



```
[ec2-user@ip-172-31-41-211 ~]$
```

From that node, let's try to find the tombstone marker for the delete. But first we need to find what the data file name is that contains vBucket #788 (or whichever one you are hunting for):

```
[ec2-user@ip-172-31-41-211 ~]$ sudo ls -al
/opt/couchbase/var/lib/couchbase/data/beer-sample | grep 788
-rw-rw----. 1 couchbase couchbase   65627 Jun  2 15:24 788.couch.1
```

In my environment the file name is '788.couch.1' so it has never been compacted. That's fine. Let's now look for the actual item 'made-on-source':

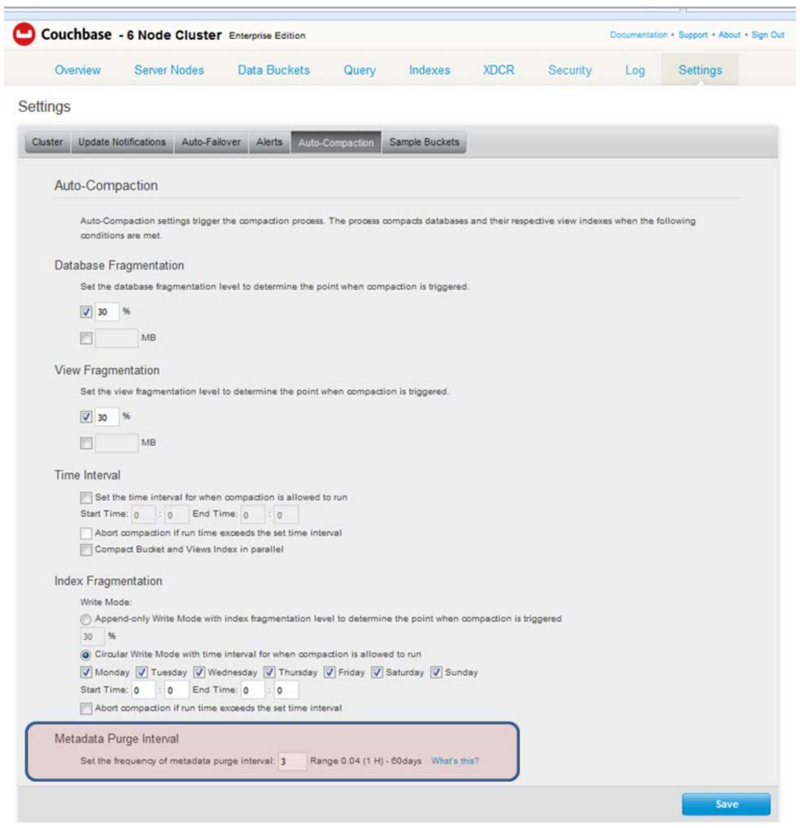
```
[ec2-user@ip-172-31-41-211 ~]$ sudo /opt/couchbase/bin/couch_dbdump
/opt/couchbase/var/lib/couchbase/data/beer-sample/788.couch.1 | grep -
B 2 -A 5 made-on-source
data: (snappy) VTKGNKUHMP
Doc seq: 246
     id: made-on-source
     rev: 2
     content_meta: 3
     size (on disk): 0
     cas: 1469446274650734592, expiry: 1469446274, flags: 0, datatype: 0,
conflict_resolution_mode: 0
     doc deleted
```

Aha! As suspected the revision count here is 2, which it was 1 on the source side. Perhaps a quick discussion of tombstones is in order...

Couchbase Server and other distributed databases maintain tombstones in order to provide eventual consistency between nodes and between clusters. Tombstones are records of expired or deleted items and they include the key for the item as well as metadata. Couchbase Server stores the key plus several bytes of metadata per deleted item. With millions of mutations, the space taken up by tombstones can grow quickly. This is especially the case if you have a large number of deletions or expired documents.

You can now configure the Metadata Purge Interval which sets how frequently a node will permanently purge metadata on deleted and expired items. This new setting will run as part of auto-compaction.

Note that you can set the metadata purge interval under the Settings screen in Couchbase. But this is just FYI. There is no need to change anything under the Settings in the Web UI.



Settings

Auto-Compaction

Auto-Compaction settings trigger the compaction process. The 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.

[✓] 30 %

[ ] 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: 0 : 0 End Time: 0 : 0

[ ] Abort compaction if run time exceeds the set time interval

[ ] Compact Bucket and Views Index in parallel

Index Fragmentation

Write Mode:

○ Append-only Write Mode with index fragmentation level to determine the point when compaction is triggered

30 %

● Circular Write Mode with time interval for when compaction is allowed to run

[✓] Monday [✓] Tuesday [✓] Wednesday [✓] Thursday [✓] Friday [✓] Saturday [✓] Sunday

Start Time: 0 : 0 End Time: 0 : 0

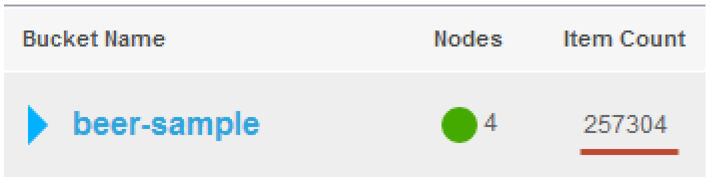
[ ] Abort compaction if run time exceeds the set time interval

Metadata Purge Interval

Set the frequency of metadata purge interval: 3 Range 0.04 (1 H) - 60days What's this?

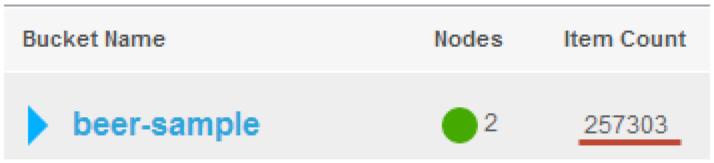
Going back to our original problem, there are still 257,304 items on the source side's beer-sample bucket and 257,303 items in the destination side's beer-sample bucket.

Source:



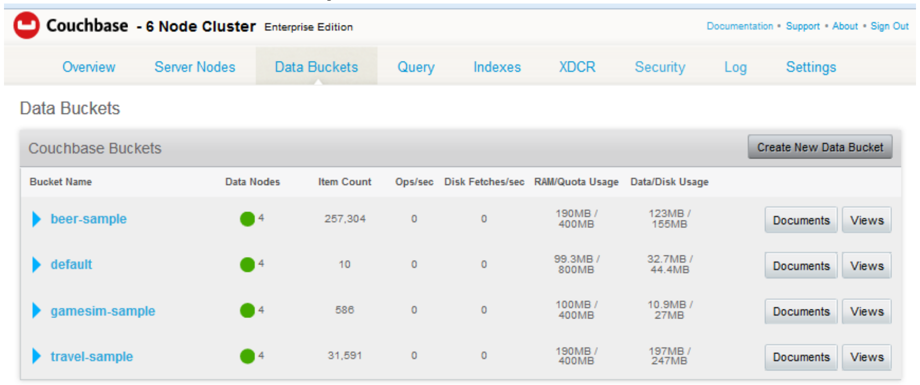
Bucket Name	Nodes	Item Count
beer-sample	● 4	257304

Destination:



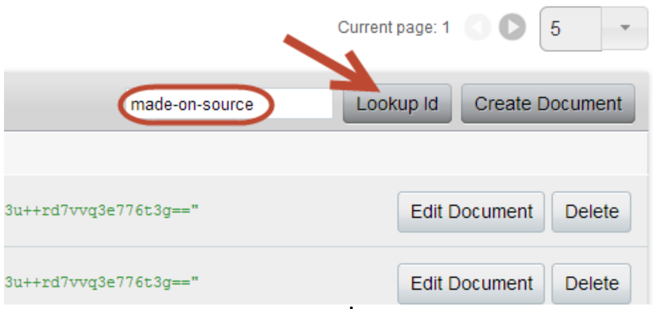
Bucket Name	Nodes	Item Count
beer-sample	● 2	257303

Switch to the Web UI browser tab for the 6-node production/source cluster and let's edit the 'made-on-source' item twice so that it has a higher revision count than the destination cluster (which has the count set to 2). Click on Data Buckets at the top, then click the Documents button for the beer-sample bucket:

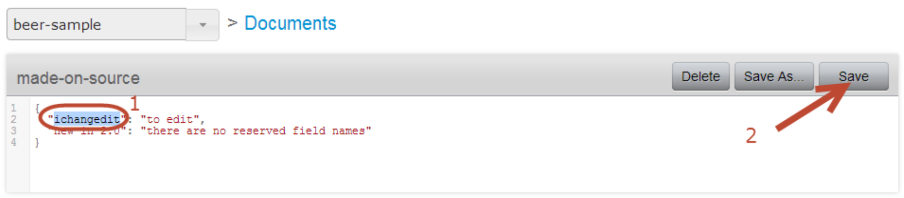


Bucket Name	Data Nodes	Item Count	Ops/sec	Disk Fetches/sec	RAM/Quota Usage	Data/Disk Usage		
beer-sample	● 4	257,304	0	0	190MB / 400MB	123MB / 155MB	Documents	Views
default	● 4	10	0	0	99.3MB / 800MB	32.7MB / 44.4MB	Documents	Views
gamesim-sample	● 4	586	0	0	100MB / 400MB	10.9MB / 27MB	Documents	Views
travel-sample	● 4	31,591	0	0	190MB / 400MB	197MB / 247MB	Documents	Views

In the search box, type in 'made-on-source' then click "Lookup id":



In the JSON file, on the 2nd line, change the word 'click' to 'ichangedit' as seen in the screenshot below then click "Save":

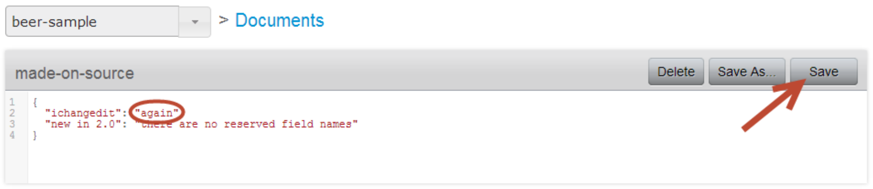


```
{
  "ichangedit": "to edit",
  "new in 2.0": "there are no reserved field names"
}
```

That was revision #2.

Now go back to terminal windows for each cluster and run "couch_dbdump syntax" and check on Source and destination clusters for rev level. Now , Source = rev:2, and Destination = rev:2 (however now it shows doc size instead of "doc deleted")

Now edit the JSON file's line #2 again. Where it says "to edit" change the words to "again" and click "Save":



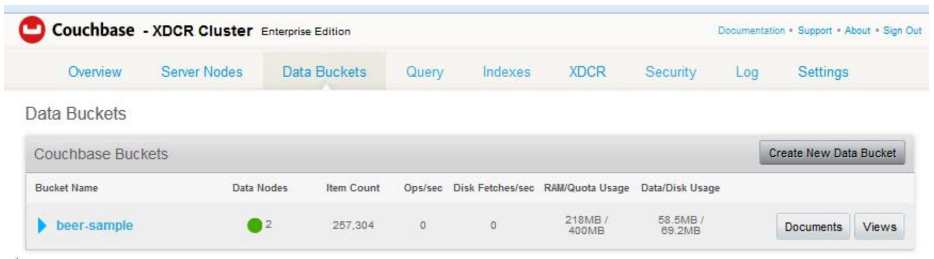
beer-sample > Documents

made-on-source

```
{
  "ichangedit": "again"
  "new in 2.0": "there are no reserved field names"
}
```

And that was revision #3.

If you switch to the browser tab for the remote/destination cluster and refresh the page for "Data Buckets", you should see the item count increased to 257,304:

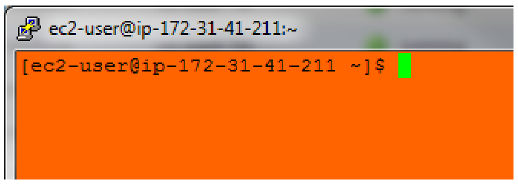


Data Buckets

Couchbase Buckets

Bucket Name	Data Nodes	Item Count	Ops/sec	Disk Fetches/sec	RAM/Quota Usage	Data/Disk Usage
beer-sample	● 2	257,304	0	0	218MB / 400MB	58.5MB / 69.2MB

Now switch to the Orange or Red node (which ever node you logged into a few pages prior when you were looking for the revision # for the item on the remote/destination side).
In my lab, that would mean XDCR Node # 2, Orange VM (but this might be red or orange for you!)



```
[ec2-user@ip-172-31-41-211 ~]$
```

From this node, let's check the revision count for the item again on the destination side. Run a command similar to the following to see the revision count and data for this key. Note the file name/vBucket # might be different for you. You essentially should just have to hit the up arrow on your keyboard and switch the –A 5 to –A 6:

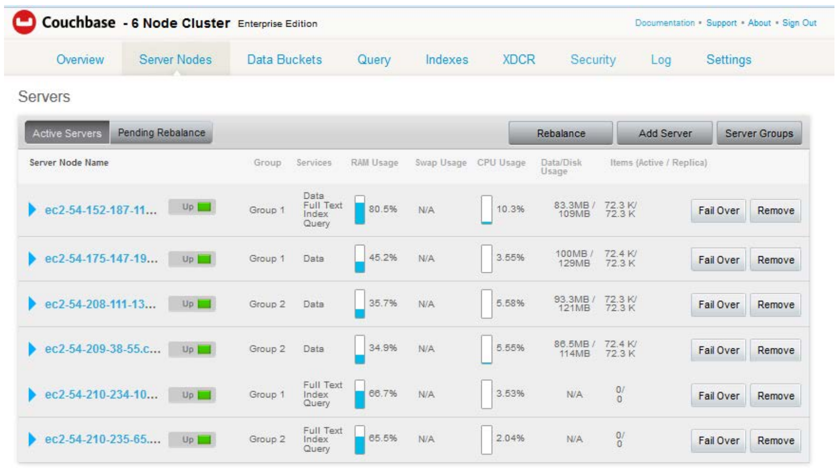
```
[ec2-user@ip-172-31-41-211 ~]$ sudo /opt/couchbase/bin/couch_dbdump
/opt/couchbase/var/lib/couchbase/data/beer-sample/788.couch.1 | grep -
B 2 -A 6 made-on-source

data: (snappy) VTKGNKUHMP
Doc seq: 247
     id: made-on-source
     rev: 3
     content_meta: 128
     size (on disk): 96
     cas: 1469449495355588608, expiry: 0, flags: 0, datatype: 1, conflict_resolution_mode: 0
     size: 87
```

Notice that the data now appears on the destination side.

Viewing outbound and inbound replication statistics:

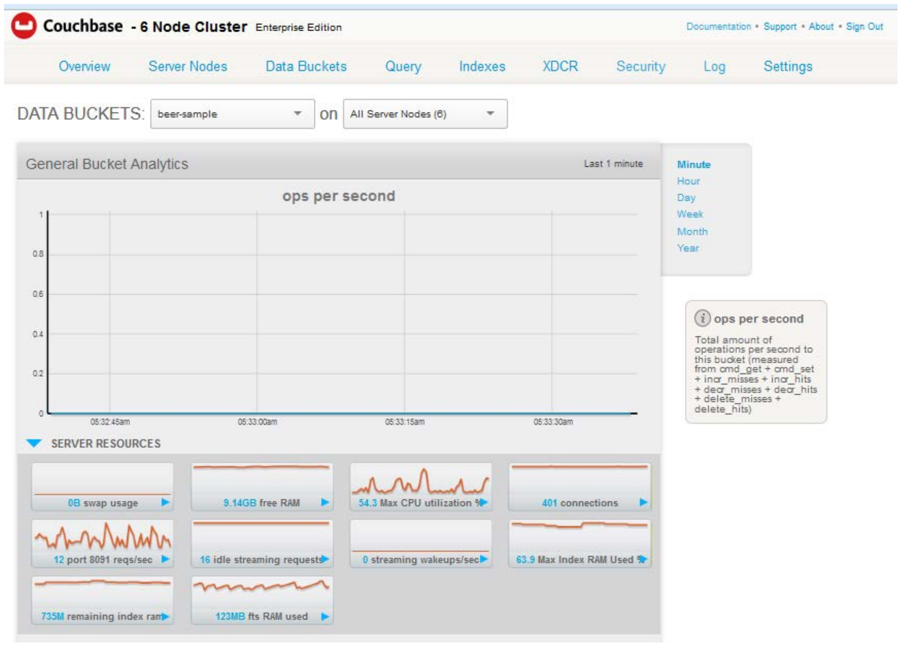
The Couchbase Web UI shows many relevant statistics for XDCR. In this section we will first load the correct web pages for outbound and inbound metrics and then use cbworkloadgen to push traffic into the production/source cluster and watch the traffic flow into the remote cluster.

Switch to the Web UI for the Production/Source 6-node cluster and click on "Server Nodes" then click on the first server in the list:



Server Node Name		Group	Services	RAM Usage	Swap Usage	CPU Usage	Data/Disk Usage	Items (Active / Replica)		
ec2-54-152-187-11...	Up	Group 1	Data Full Text Index Query	80.5%	N/A	10.3%	83.3MB / 109MB	72.3 K/ 72.3 K	Fail Over	Remove
ec2-54-175-147-19...	Up	Group 1	Data	45.2%	N/A	3.55%	100MB / 129MB	72.4 K/ 72.3 K	Fail Over	Remove
ec2-54-208-111-13...	Up	Group 2	Data	35.7%	N/A	5.58%	93.3MB / 121MB	72.3 K/ 72.3 K	Fail Over	Remove
ec2-54-209-38-55.c...	Up	Group 2	Data	34.9%	N/A	5.55%	86.5MB / 114MB	72.4 K/ 72.3 K	Fail Over	Remove
ec2-54-210-234-10...	Up	Group 1	Full Text Index Query	66.7%	N/A	3.53%	N/A	0/ 0	Fail Over	Remove
ec2-54-210-235-65....	Up	Group 2	Full Text Index Query	65.5%	N/A	2.04%	N/A	0/ 0	Fail Over	Remove

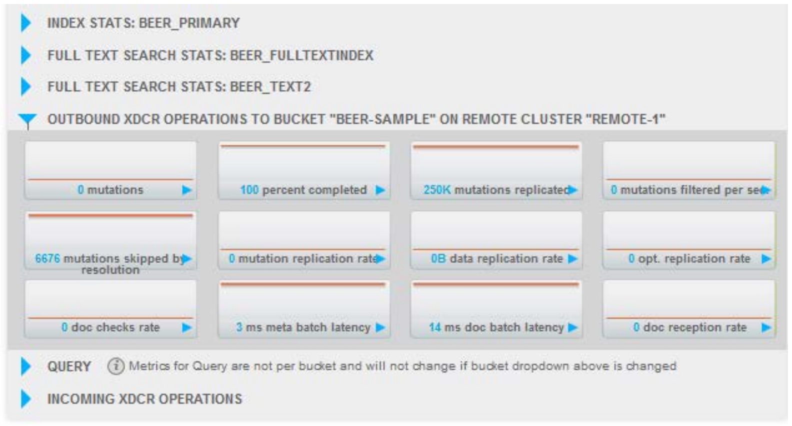
On the metrics page, switch to the beer-sample bucket on All Server Nodes (6):



ops per second

Total amount of operations per second to this bucket (measured from cmd_get + cmd_set + incr_misses + incr_hits + decr_misses + decr_hits + delete_misses + delete_hits)

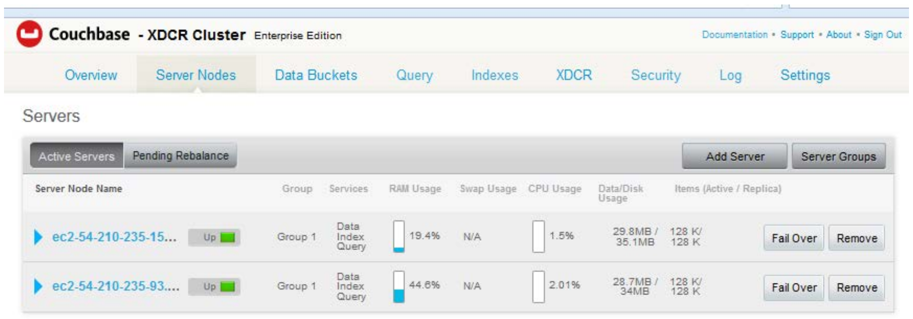
Scroll down and click the relevant blue arrow to expand the "OUTBOUND XDCR OPERATIONS" section:


INDEX STATS: BEER_PRIMARY
FULL TEXT SEARCH STATS: BEER_FULLTEXTINDEX
FULL TEXT SEARCH STATS: BEER_TEXT2
OUTBOUND XDCR OPERATIONS TO BUCKET "BEER-SAMPLE" ON REMOTE CLUSTER "REMOTE-1"
0 mutations
100 percent completed
250K mutations replicated
0 mutations filtered per sec
6676 mutations skipped by resolution
0 mutation replication rate
0B data replication rate
0 opt. replication rate
0 doc checks rate
3 ms meta batch latency
14 ms doc batch latency
0 doc reception rate
QUERY
INCOMING XDCR OPERATIONS

Metrics for Query are not per bucket and will not change if bucket dropdown above is changed

You should see 0 outbound XDCR mutations here as there is no data currently being written to, updated or deleted on the source/production side.
Keep this tab as is b/c we will return to it once we start generating traffic.
Next we need to also open the INBOUND XDCR metrics on the remote/destination side.

Switch to the Web UI for the remote/destination 2-node cluster and click on "Server Nodes" then click on the first server in the list:

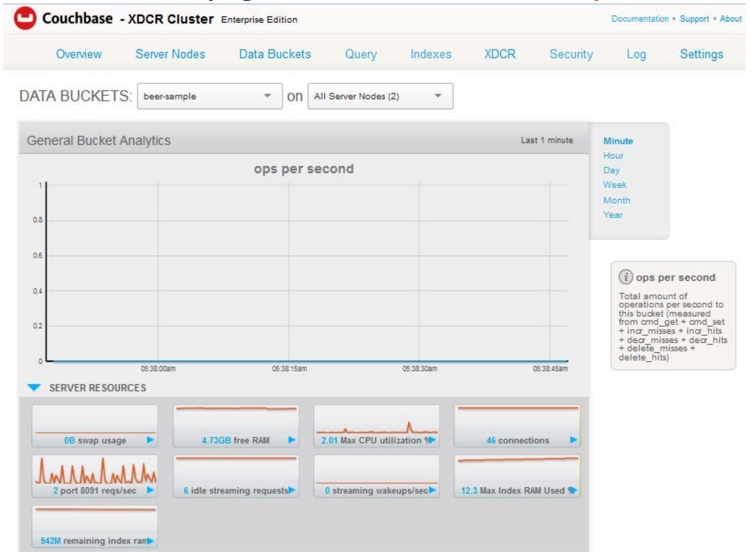


Couchbase - XDCR Cluster Enterprise Edition

Servers

Server Node Name		Group	Services	RAM Usage	Swap Usage	CPU Usage	Data/Disk Usage	Items (Active / Replica)
ec2-54-210-235-15...	Up	Group 1	Data Index Query	19.4%	N/A	1.5%	29.8MB / 35.1MB	128 K/ 128 K
ec2-54-210-235-93....	Up	Group 1	Data Index Query	44.6%	N/A	2.01%	28.7MB / 34MB	128 K/ 128 K

On the metrics page, switch to the beer-sample bucket on All Server Nodes (2):



Couchbase - XDCR Cluster Enterprise Edition

DATA BUCKETS: beer-sample on All Server Nodes (2)

General Bucket Analytics — Last 1 minute

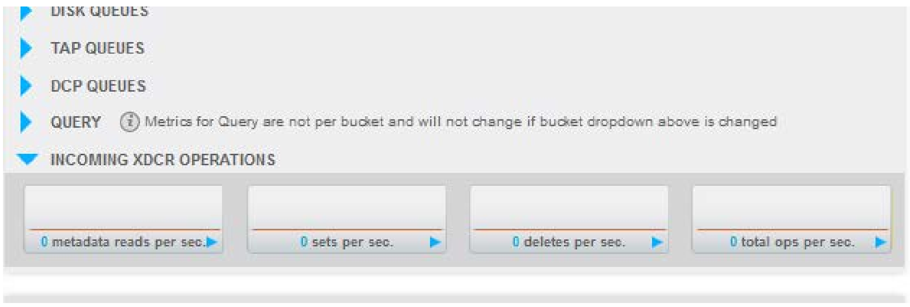
ops per second

ops per second: Total amount of operations per second to this bucket (measured from cmd_get + cmd_set + incr_misses + incr_hits + decr_misses + decr_hits + delete_misses + delete_hits)

SERVER RESOURCES

0B swap usage; 4.73GB free RAM; 2.01 Max CPU utilization %; 46 connections; 2 port 8091 reqs/sec; 6 idle streaming requests; 0 streaming wakeups/sec; 12.3 Max Index RAM Used %; 942M remaining index ram

Scroll down and click the relevant blue arrow to expand the "INCOMING XDCR OPERATIONS" section:



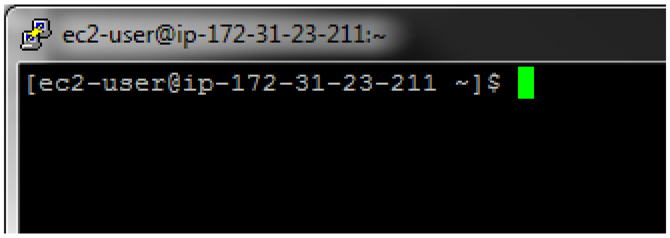
DISK QUEUES
TAP QUEUES
DCP QUEUES
QUERY — Metrics for Query are not per bucket and will not change if bucket dropdown above is changed
INCOMING XDCR OPERATIONS

0 metadata reads per sec.; 0 sets per sec.; 0 deletes per sec.; 0 total ops per sec.

This section should also show 0 for all 4 metrics it displays.

Now we are ready to dump some fake data into the 6-node production/source cluster's beer-sample bucket!

Switch to the App Server (black VM):

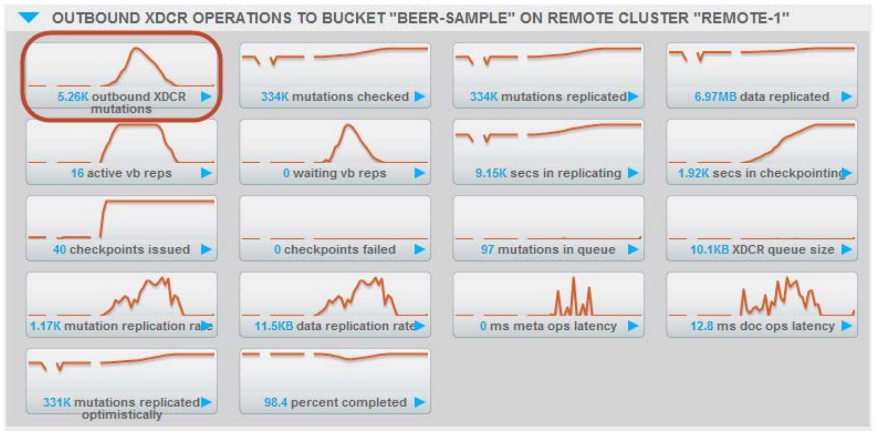


```
ec2-user@ip-172-31-23-211:~
[ec2-user@ip-172-31-23-211 ~]$
```

Run the following command to insert 2 million keys (2,000,000 keys) of size 10 bytes with 100% write workload into the source beer-sample bucket. Use Node #1 (dark blue VM) in the source/production cluster's public hostname for the command below (WARNING: Do not copy+paste the command below as the ASCII characters will NOT translate correct and the data will go incorrectly into the default bucket):

```
[ec2-user@ip-172-31-23-211 ~]$ cbworkloadgen -n ec2-54-85-43-
x.compute-1.amazonaws.com:8091 -u Administrator -p couchbase –b beer-
sample -i 2000000 -r 1 -s 10
  [#                  ] 5.6% (39000/estimated 700000 msgs)
```

Switch to the 6-node source/production cluster's Web UI tab and observe the activity:

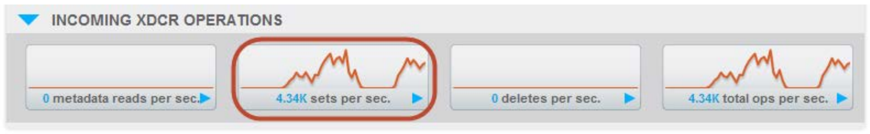


OUTBOUND XDCR OPERATIONS TO BUCKET "BEER-SAMPLE" ON REMOTE CLUSTER "REMOTE-1"

5.26K outbound XDCR mutations
334K mutations checked
334K mutations replicated
6.97MB data replicated
16 active vb reps
0 waiting vb reps
9.15K secs in replicating
1.92K secs in checkpointing
40 checkpoints issued
0 checkpoints failed
97 mutations in queue
10.1KB XDCR queue size
1.17K mutation replication rate
11.5KB data replication rate
0 ms meta ops latency
12.8 ms doc ops latency
331K mutations replicated optimistically
98.4 percent completed

Notice in the screenshot above that my source cluster is pushing out about 5,260 (5.26K) mutations outbound. Yours will vary.

Switch to the 2-node destination cluster's Web UI and observe the activity here:



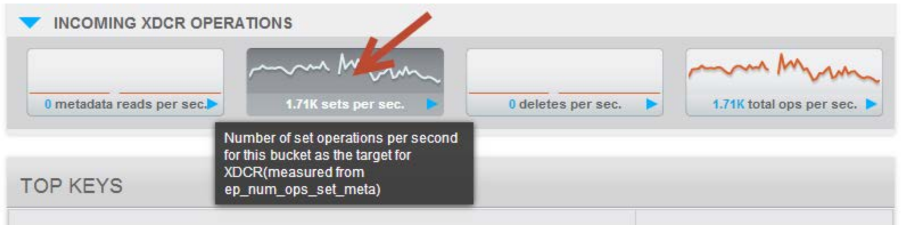
INCOMING XDCR OPERATIONS

0 metadata reads per sec.
4.34K sets per sec.
0 deletes per sec.
4.34K total ops per sec.

Notice the destination cluster is seeing about 4,340 (4.34K) sets per second.

Note that cbworkloadgen will take about 5 - 7 minutes to insert the 2 million keys. So, while that is occurring, I suggest clicking on some of the graphs in the source and destination cluster's web UI to see what each chart means.

For example, here I clicked on a specific graph and pulled up an explanation of what the set operations per second chart actually means:



Number of set operations per second for this bucket as the target for XDCR(measured from ep_num_ops_set_meta)

When you are finished studying the XDCR metrics, switch back to the App Server (black VM) and if the cbworkloadgen is still running just hit <CTRL> + C in the PuTTY shell to stop the workload generator. If the generator stopped on its own b/c it already inserted 2 million keys, that is fine as well, no need to terminate it.(most likely it will run bucket out of memory)

```
2014-06-02 17:37:34,796: s0 backing off, secs: 10.0
2014-06-02 17:37:45,113: s0 backing off, secs: 10.0
2014-06-02 17:37:55,389: s0 backing off, secs: 10.0
2014-06-02 17:38:05,595: s0 backing off, secs: 10.0
<CTRL> + C
^Cinterrupted.
[ec2-user@ip-172-31-23-211 ~]$
```

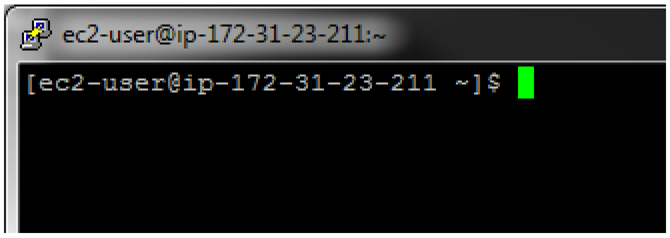
The specific error messages that you see above mean that the memory on the source cluster is too full and the nodes are sending 'tmp_oom' messages.

You can ignore these errors.

Viewing internal XDCR settings via REST API:

There are internal settings for XDCR which are only exposed via the REST API. These settings will change the replication behavior, performance, and timing.

Switch to the App Server (black VM):



```
[ec2-user@ip-172-31-23-211 ~]$
```

Run the following curl command against the public hostname of Node #1 on the 6-node production/source side:

```
[ec2-user@ip-172-31-23-211 ~]$ curl -u Administrator:couchbase
http://ec2-54-85-43-x.compute-1.amazonaws.com:8091/internalSettings
```

```
{"indexAwareRebalanceDisabled":false,"rebalanceIndexWaitingDisabled":false,"rebalanceIndexPausing
Disabled":false,"rebalanceIgnoreViewCompactions":false,"rebalanceMovesPerNode":1,"rebalanceMovesB
eforeCompaction":64,"maxParallelIndexers":4,"maxParallelReplicaIndexers":2,"maxBucketCount":10,"x
dcrMaxConcurrentReps":32,"xdcrCheckpointInterval":1800,"xdcrWorkerBatchSize":500,"xdcrDocBatchSiz
eKb":2048,"xdcrFailureRestartInterval":30,"xdcrOptimisticReplicationThreshold":256,"restRequestLi
mit":"","capiRequestLimit":"","dropRequestMemoryThresholdMiB":""}
```

I have highlighted the XDCR specific settings in the JSON file above using multiple colors. The above highlighted settings are applied to all replications for this cluster.

The next command will also display global settings for all replications for the cluster (Notice that some of these settings are the same as the command output above):

```
[ec2-user@ip-172-31-23-211 ~]$ curl -u Administrator:couchbase
http://ec2-54-85-43-x.compute-
1.amazonaws.com:8091/settings/replications/
```

```
{"checkpointInterval":1800,"docBatchSizeKb":2048,"failureRestartInterval":10,"logLevel":"Info","o
ptimisticReplicationThreshold":256,"sourceNozzlePerNode":2,"statsInterval":1000,"targetNozzlePerN
ode":2,"workerBatchSize":500}
```

Finally, it is also possible to retrieve the settings for a specific replication for a bucket. But to do this, we first have to get the replication ID for the beer-sample -> beer-sample replication.

Run the following curl command against the Node #1 public hostname in the 6-node source cluster:

```
[ec2-user@ip-172-31-23-211 ~]$ curl -u Administrator:couchbase
http://ec2-54-85-43-128.compute-
1.amazonaws.com:8091/pools/default/tasks
```

```
[{"type":"rebalance","status":"notRunning"},{"cancelURI":"/controller/cancelXDCR/2d1e70ee1689ff1c
bdcb45b93891fcaa%2Fbeer-sample%2Fbeer-
sample","settingsURI":"/settings/replications/2d1e70ee1689ff1cbdcb45b93891fcaa%2Fbeer-
sample%2Fbeer-
sample","status":"running","replicationType":"xmem","id":"2d1e70ee1689ff1cbdcb45b93891fcaa/beer-
sample/beer-sample","source":"beer-
sample","target":"/remoteClusters/2d1e70ee1689ff1cbdcb45b93891fcaa/buckets/beer-
sample","continuous":true,"type":"xdcr","recommendedRefreshPeriod":10,"changesLeft":0,"docsChecke
d":1461872,"docsWritten":1461872,"errors":[]}]
```

Notice that the replication ID is highlighted in yellow in the output above. You will need this specific ID from YOUR output for the next command. Also notice that this replication ID has checked and written 1,461,872 items (the specific # you see may vary).

Use the replication ID to run a curl command to get the specific settings for that replication ID:

```
[ec2-user@ip-172-31-23-211 ~]$ curl -u Administrator:couchbase
http://43-128.compute-
1.amazonaws.com:8091/settings/replications/2d1e70ee1689ff1cbdcb45b9389
1fcaa%2Fbeer-sample%2Fbeer-sample
{"checkpointInterval":1800,"docBatchSizeKb":2048,"failureRestartInterval":10,"filterExpression":"
","logLevel":"Info","optimisticReplicationThreshold":256,"pauseRequested":false,"sourceNozzlePerN
ode":2,"statsInterval":1000,"targetNozzlePerNode":2,"type":"xmem","workerBatchSize":500}
```

More details about how to change the advanced settings can be found here:
http://docs.couchbase.com/admin/admin/Tasks/xdcr-specify-adv-settings.html

Configuring encrypted, bidirectional, optimistic replication between 2 clusters:

Bidirectional Replication

In this section of the lab, we will create a bidirectional replication between the production 6-node and remote 2-node cluster.

Replication is normally unidirectional from one cluster to another. Bidirectional Replication allows two clusters to replicate data with each other. Setting up bidirectional replication in Couchbase Server involves setting up two unidirectional replication links from one cluster to the other. This is useful when you want to load balance your workload across two clusters where each cluster bi-directionally replicates data to the other cluster.

To configure bidirectional replication between two clusters, you need to provide settings for two separate replication streams. One stream replicates changes from Cluster A to Cluster B, another stream replicates changes from Cluster B to Cluster A. To configure a bidirectional replication:

Create a replication from Cluster A to Cluster B on Cluster A.

Create a replication from Cluster B to Cluster A on Cluster B.

You do not need identical topologies for both clusters; you can have a different number of nodes in each cluster, and different RAM and persistence configurations.

Optimistic Replication

The other setting we will alter in this section is the optimistic replication threshold which is set to 256 bytes by default. The setting can be between 0 bytes to 20 MB (2097152 bytes). This setting can improve the latency for XDCR. An explanation for why follows, but first you should understand how XDCR typically works.

Typically XDCR on the source cluster will get metadata for documents larger than this size (256 bytes by default) from the remote cluster once before replicating the document to a destination cluster. One of the reasons for this is to verify the revision count of the item in the destination cluster versus the revision count for the same item in the source cluster. If the revision count is higher in the destination cluster, there is no need to send the item to the destination as it would be discarded anyway. If the source item's revision count is higher, then the item is placed into the replication queue. Once the item arrives at the destination cluster, the revision count from the metadata in the destination cluster is queried again to make sure that a change did not occur on the destination cluster while the item was traveling over the network from the source to the destination. So, then a node on the destination side also verifies that this incoming item has a higher revision count than the item in the destination cluster and only then does the incoming item overwrite the item in the destination cluster.

However, when a document is smaller than the number of bytes provided as this parameter, XDCR on the source side immediately puts it into the replication queue without getting metadata from the destination side to the source cluster.

If the document is deleted on a source cluster, XDCR source will not fetch metadata for the document from the destination before it sends this update to a destination cluster. Once a document reaches the destination cluster, XDCR will fetch the metadata from the local destination cluster and perform conflict resolution between documents. If the document 'loses' conflict resolution, Couchbase Server discards it on the destination cluster and keeps the version on the destination. This new feature improves replication latency, particularly when you replicate small documents.

There are tradeoffs when you change this setting. If you set this low relative to document size, XDCR on the source will frequently check metadata from the destination. This will increase latency during replication, it also means that it will get metadata from destination cluster before it puts a document into the replication queue, and will get it again for the destination to perform conflict resolution. The advantage is that you do not waste network bandwidth since XDCR will send less documents that will 'lose' the conflict resolution.

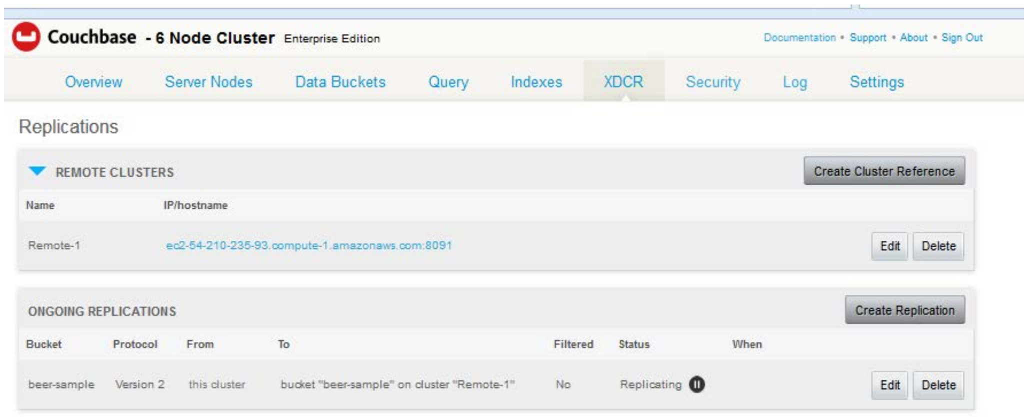
If you set this very high relative to document size, XDCR source cluster will fetch less metadata from the destination cluster which will improve latency during replication. This also means that you will increase the rate at which XDCR puts items immediately into the replication queue which can potentially overwhelm your network, especially if you set a high number of parallel replicators. This may increase the number of documents sent by XDCR which ultimately 'lose' conflicts at the destination which wastes network bandwidth.

So how to configure the setting for optimistic replication depends on your specific use case the insert/update/delete traffic patterns.

Note: XDCR does not fetch metadata for documents that are deleted.

First let's delete the existing ongoing unidirectional replication so we can get a fresh start. Then we will delete the beer-sample and default buckets on both clusters. Then finally we will set up a fresh bucket on both sides for bidirectional replication and then study the optimistic replication threshold.

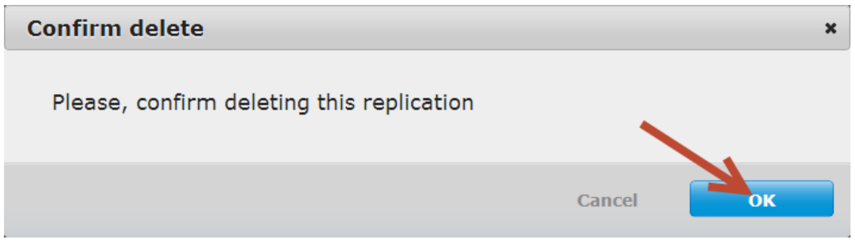
Switch to the browser tab for the 6-node production cluster and click on XDCR. Then click Delete next to the beer-sample replication:



Name	IP/hostname
Remote-1	ec2-54-210-235-93.compute-1.amazonaws.com:8091

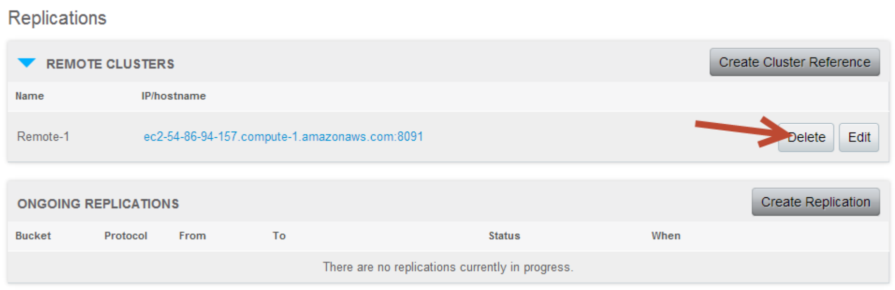
Bucket	Protocol	From	To	Filtered	Status	When
beer-sample	Version 2	this cluster	bucket "beer-sample" on cluster "Remote-1"	No	Replicating	

Click OK on the pop-up:


Confirm delete

Please, confirm deleting this replication

In the same Web UI, click Delete next to Remote-1 to delete the remote cluster as well:


Replications

REMOTE CLUSTERS

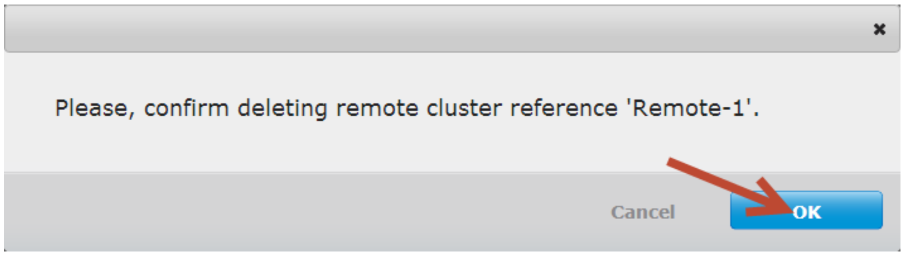
Name	IP/hostname
Remote-1	ec2-54-86-94-157.compute-1.amazonaws.com:8091

ONGOING REPLICATIONS

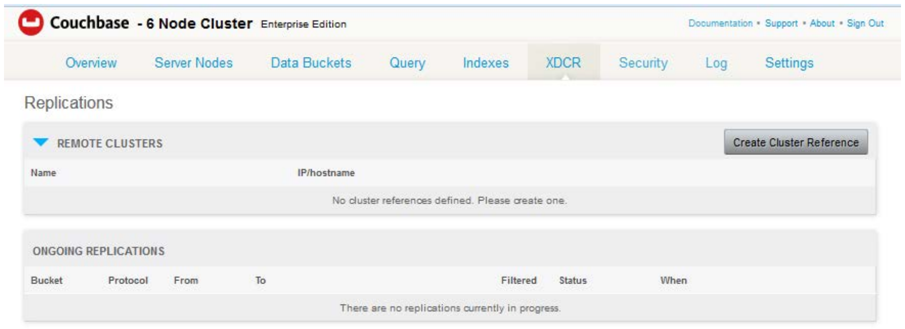
Bucket	Protocol	From	To	Status	When

There are no replications currently in progress.

Confirm the deletion by clicking OK:


Please, confirm deleting remote cluster reference 'Remote-1'.

You should now see no replication settings defined for the 6-node cluster:


Couchbase - 6 Node Cluster Enterprise Edition

Replications

REMOTE CLUSTERS

Name	IP/hostname

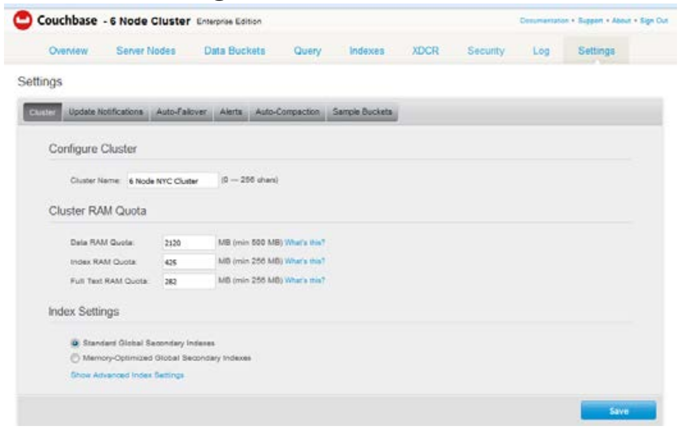
No cluster references defined. Please create one.

ONGOING REPLICATIONS

Bucket	Protocol	From	To	Filtered	Status	When

There are no replications currently in progress.

Go to the settings tab and rename the cluster "6 node NYC cluster"


Couchbase - 6 Node Cluster Enterprise Edition

Settings

Configure Cluster

Cluster Name: 6 Node NYC Cluster (0 — 256 chars)

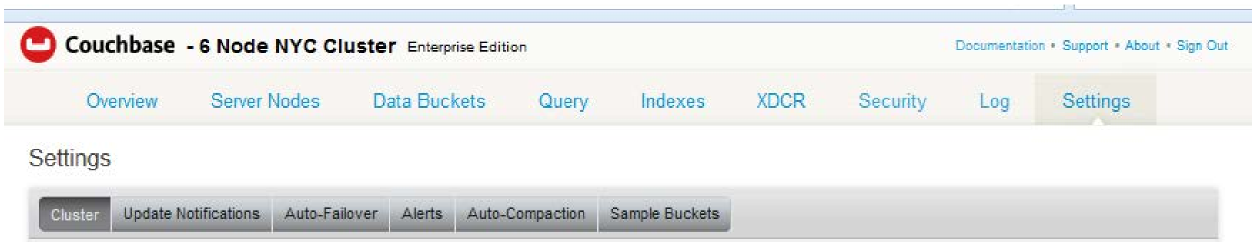
Cluster RAM Quota

Data RAM Quota: 2120 MB (min 500 MB) What's this?
Index RAM Quota: 425 MB (min 256 MB) What's this?
Full Text RAM Quota: 282 MB (min 256 MB) What's this?

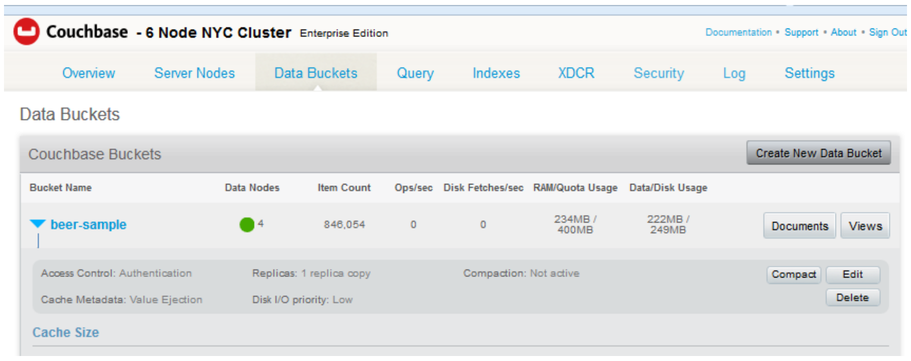
Index Settings

[✓ Standard Global Secondary Indexes]
Memory-Optimized Global Secondary Indexes

Show Advanced Index Settings


Couchbase - 6 Node NYC Cluster Enterprise Edition
Documentation • Support • About • Sign Out
Overview Server Nodes Data Buckets Query Indexes XDCR Security Log Settings
Settings
Cluster Update Notifications Auto-Failover Alerts Auto-Compaction Sample Buckets

Next, in the same 6-node cluster's UI, click on Data Buckets at the top and then expand the beer-sample bucket by clicking on the blue arrow next to it and finally click on Delete:

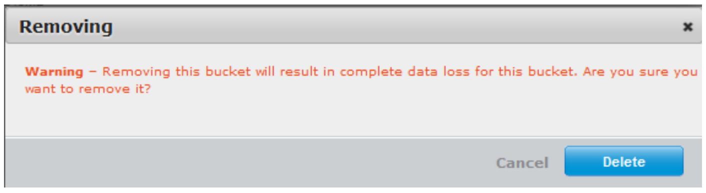

Couchbase - 6 Node NYC Cluster Enterprise Edition
Documentation • Support • About • Sign Out
Overview Server Nodes Data Buckets Query Indexes XDCR Security Log Settings
Data Buckets
Couchbase Buckets
Create New Data Bucket

Bucket Name	Data Nodes	Item Count	Ops/sec	Disk Fetches/sec	RAM/Quota Usage	Data/Disk Usage
beer-sample	● 4	846,054	0	0	234MB / 400MB	222MB / 249MB

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

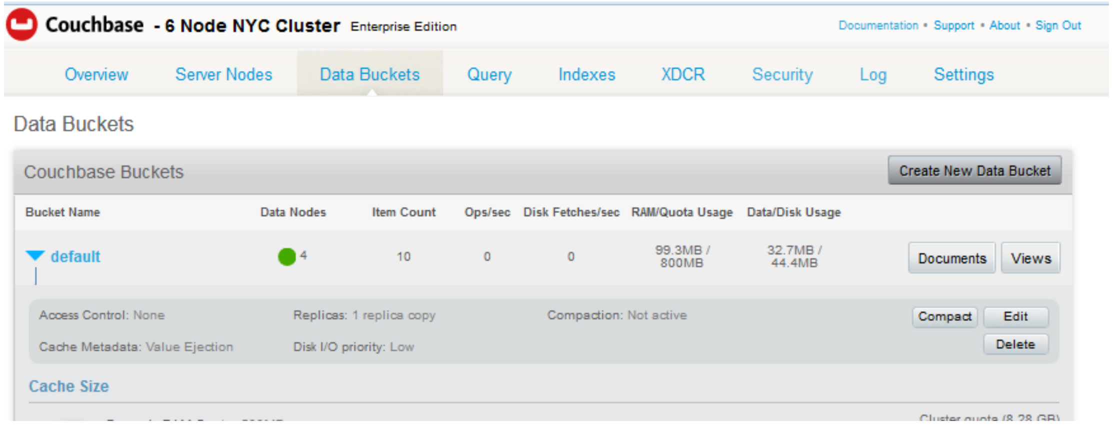
Documents Views
Compact Edit
Delete
Cache Size

Click Delete to confirm the removal:



Warning – Removing this bucket will result in complete data loss for this bucket. Are you sure you want to remove it?

The beer-sample bucket will now disappear from the 6-node cluster's UI. Next expand the 'default' bucket's settings by clicking the blue arrow next to 'default' and then click Delete:

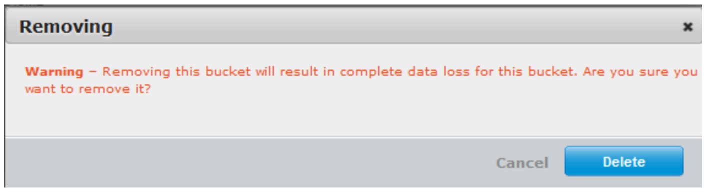

Couchbase - 6 Node NYC Cluster Enterprise Edition
Documentation • Support • About • Sign Out
Overview Server Nodes Data Buckets Query Indexes XDCR Security Log Settings
Data Buckets
Couchbase Buckets
Create New Data Bucket

Bucket Name	Data Nodes	Item Count	Ops/sec	Disk Fetches/sec	RAM/Quota Usage	Data/Disk Usage
default	● 4	10	0	0	99.3MB / 800MB	32.7MB / 44.4MB

Access Control: None Replicas: 1 replica copy Compaction: Not active
Cache Metadata: Value Ejection Disk I/O priority: Low

Documents Views
Compact Edit
Delete
Cache Size

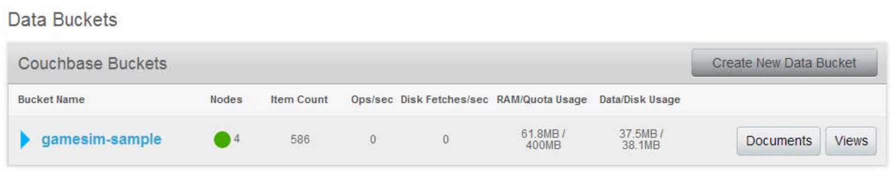
Scroll down on the 'Configure Bucket' pop-up and click Delete:


Removing

Warning – Removing this bucket will result in complete data loss for this bucket. Are you sure you want to remove it?

Next delete the travel-sample bucket.

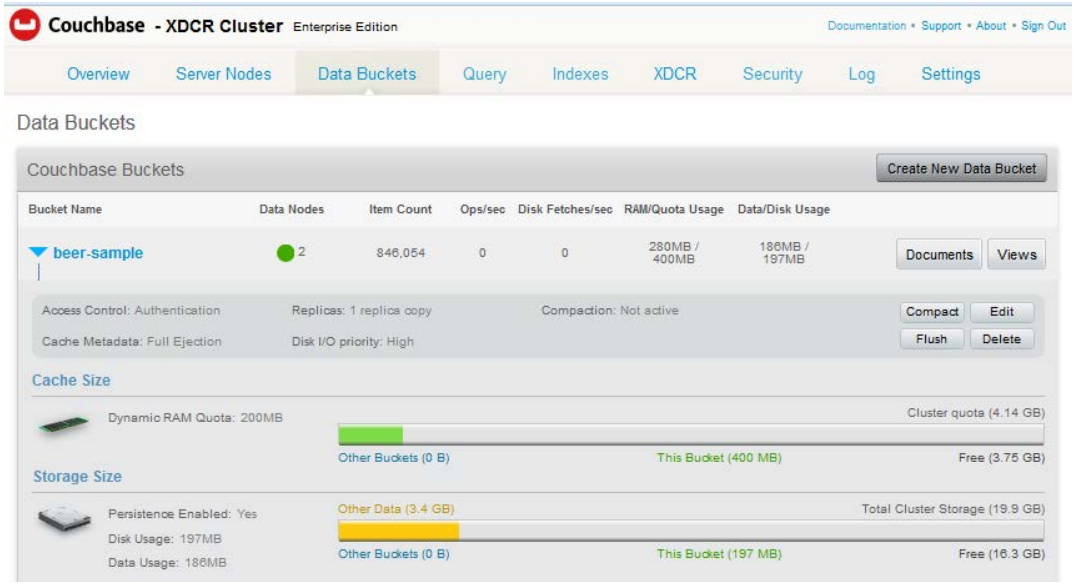
You should now see only the gamesim-sample bucket. Leave this bucket alone:


Data Buckets

Couchbase Buckets

Bucket Name	Nodes	Item Count	Ops/sec	Disk Fetches/sec	RAM/Quota Usage	Data/Disk Usage
gamesim-sample	● 4	586	0	0	61.8MB / 400MB	37.5MB / 38.1MB

Now we'll delete the beer-sample bucket on the remote side. Switch to the remote 2-node cluster's Web UI and click on Data Buckets at the top, then expand the settings for beer-sample by click the blue arrow next to beer-sample and finally click on Delete:


Couchbase - XDCR Cluster Enterprise Edition
Documentation • Support • About • Sign Out
Overview
Server Nodes
Data Buckets
Query
Indexes
XDCR
Security
Log
Settings

Data Buckets

Couchbase Buckets

Bucket Name	Data Nodes	Item Count	Ops/sec	Disk Fetches/sec	RAM/Quota Usage	Data/Disk Usage
beer-sample	● 2	846,054	0	0	280MB / 400MB	186MB / 197MB

Access Control: Authentication
Replicas: 1 replica copy
Compaction: Not active
Cache Metadata: Full Ejection
Disk I/O priority: High

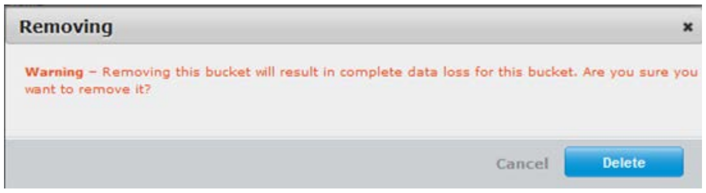
Cache Size

Dynamic RAM Quota: 200MB
Cluster quota (4.14 GB)
Other Buckets (0 B) This Bucket (400 MB) Free (3.75 GB)

Storage Size

Persistence Enabled: Yes
Disk Usage: 197MB
Data Usage: 186MB
Other Data (3.4 GB) Total Cluster Storage (19.9 GB)
Other Buckets (0 B) This Bucket (197 MB) Free (16.3 GB)

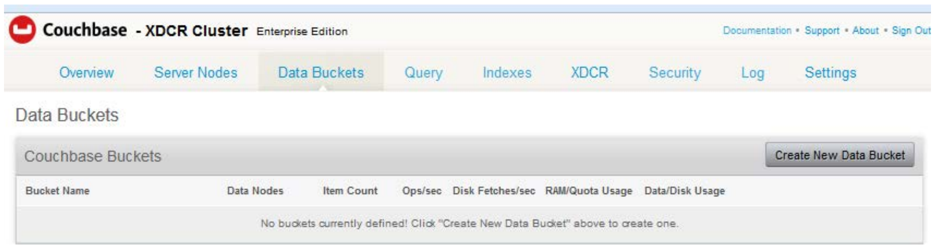
Click Delete to confirm the removal:



Removing

Warning – Removing this bucket will result in complete data loss for this bucket. Are you sure you want to remove it?

You should now see zero buckets on the remote side's Web UI.
Click on "Create New Data Bucket":

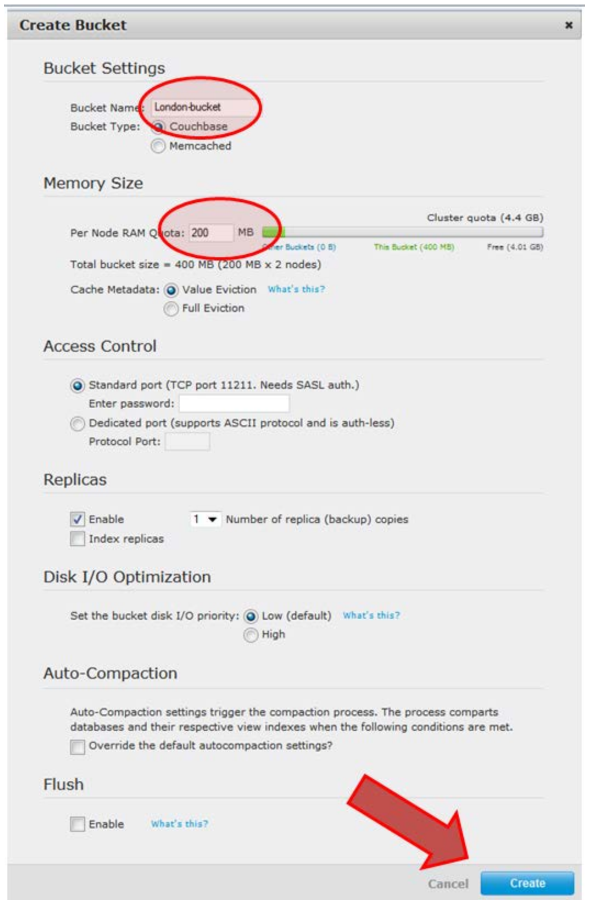


Bucket Name	Data Nodes	Item Count	Ops/sec	Disk Fetches/sec	RAM/Quota Usage	Data/Disk Usage

No buckets currently defined! Click "Create New Data Bucket" above to create one.

On the Create Bucket pop-up, use the following settings. Note only the settings in Red need to be changed:

- **Bucket Name: London-bucket**
- **Bucket Type: Couchbase**
- **Per Node RAM Quota: 200 MB**
- **Cache Metadata: Full Eviction**
- **Standard port: selected and no password**
- **Replicas: Enabled and set to 1**
- **Index replicas: Not checked**
- **Disk I/O optimization: Low**
- **Override the default autocompaction settings: Not checked**
- **Flush Enable: Not checked**
- **Click Create**


Create Bucket

Bucket Settings

Bucket Name: London-bucket
Bucket Type: [✓ Couchbase] Memcached

Memory Size

Per Node RAM Quota: 200 MB Cluster quota (4.4 GB)
Total bucket size = 400 MB (200 MB x 2 nodes)
Cache Metadata: [✓ Value Eviction] Full Eviction What's this?

Access Control

[✓ Standard port (TCP port 11211. Needs SASL auth.)]
Enter password:
Dedicated port (supports ASCII protocol and is auth-less)
Protocol Port:

Replicas

[✓ Enable] 1 Number of replica (backup) copies
Index replicas

Disk I/O Optimization

Set the bucket disk I/O priority: [✓ Low (default)] High What's this?

Auto-Compaction

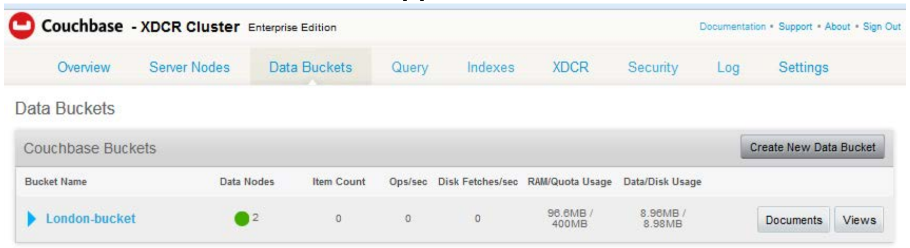
Auto-Compaction settings trigger the compaction process. The process comparts databases and their respective view indexes when the following conditions are met.
Override the default autocompaction settings?

Flush

Enable What's this?

Cancel Create

The London bucket will now appear in the UI:


Couchbase - XDCR Cluster Enterprise Edition

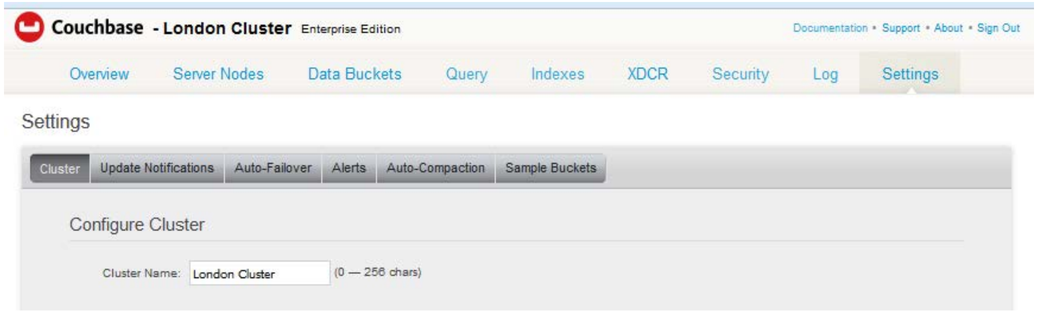
Overview Server Nodes Data Buckets Query Indexes XDCR Security Log Settings

Data Buckets

Couchbase Buckets Create New Data Bucket

Bucket Name	Data Nodes	Item Count	Ops/sec	Disk Fetches/sec	RAM/Quota Usage	Data/Disk Usage		
London-bucket	● 2	0	0	0	96.6MB / 400MB	8.96MB / 8.98MB	Documents	Views

Lastly go to the "Settings" tab and rename the cluster "London Cluster".


Couchbase - London Cluster Enterprise Edition

Overview Server Nodes Data Buckets Query Indexes XDCR Security Log Settings

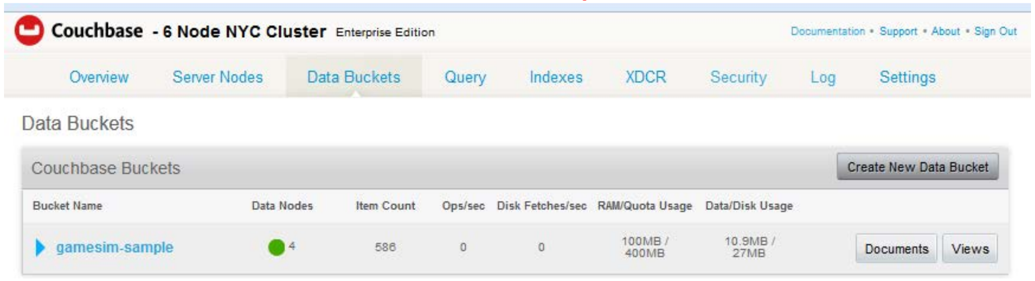
Settings

Cluster Update Notifications Auto-Failover Alerts Auto-Compaction Sample Buckets

Configure Cluster

Cluster Name: London Cluster (0 — 256 chars)

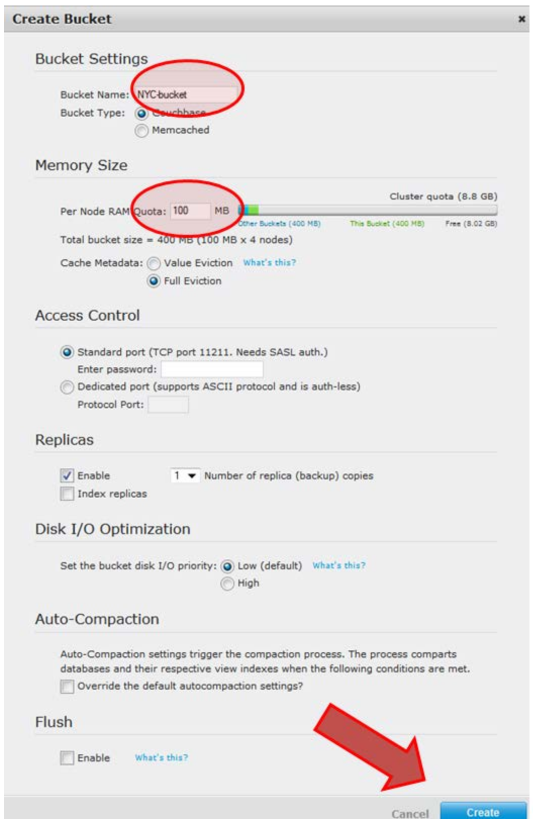
Now, switch back to the 6-node production cluster and let's create a bucket there named NYC-bucket. Click on Data Buckets at the top and then click "Create new Data Bucket":



Bucket Name	Data Nodes	Item Count	Ops/sec	Disk Fetches/sec	RAM/Quota Usage	Data/Disk Usage
gamesim-sample	● 4	586	0	0	100MB / 400MB	10.9MB / 27MB

On the Create Bucket pop-up, use the following settings. Note only the settings in Red need to be changed:

Bucket Name: NYC-bucket
Bucket Type: Couchbase
Per Node RAM Quota: 100 MB
CacheMetadata: Full Eviction
Standard port: selected and no password
Replicas: Enabled and set to 1
Index replicas: Not checked
Disk I/O optimization: low
Override the default autocompaction settings: Not checked
Flush Enable: Not checked
Click Create



Create Bucket

Bucket Settings

Bucket Name: NYC-bucket
Bucket Type: [✓ Couchbase] Memcached

Memory Size

Per Node RAM Quota: 100 MB — Cluster quota (8.8 GB) — Other Buckets (400 MB) — This Bucket (400 MB) — Free (8.02 GB)
Total bucket size = 400 MB (100 MB x 4 nodes)
Cache Metadata: Value Eviction [✓ Full Eviction] What's this?

Access Control

[✓ Standard port (TCP port 11211. Needs SASL auth.)]
Enter password:
Dedicated port (supports ASCII protocol and is auth-less)
Protocol Port:

Replicas

[✓ Enable] 1 Number of replica (backup) copies
Index replicas

Disk I/O Optimization

Set the bucket disk I/O priority: [✓ Low (default)] High What's this?

Auto-Compaction

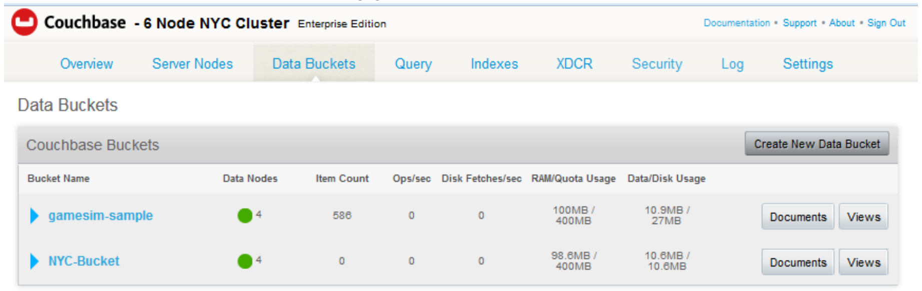
Auto-Compaction settings trigger the compaction process. The process comparts databases and their respective view indexes when the following conditions are met.
Override the default autocompaction settings?

Flush

Enable What's this?

Cancel Create

The NYC bucket will now appear in the UI:



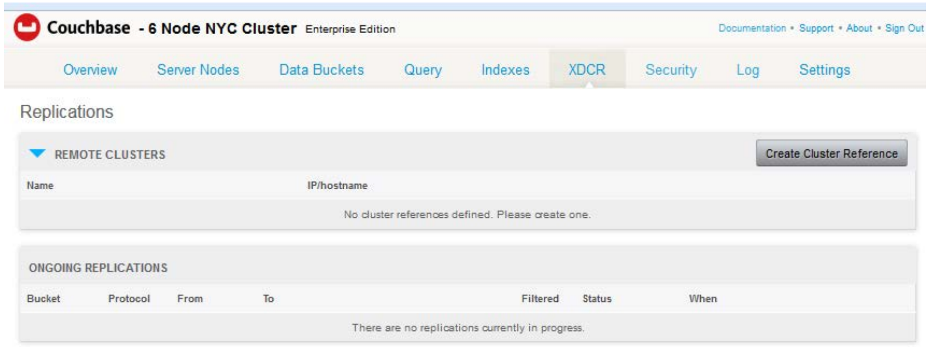
Couchbase - 6 Node NYC Cluster Enterprise Edition

Data Buckets

Couchbase Buckets — Create New Data Bucket

Bucket Name	Data Nodes	Item Count	Ops/sec	Disk Fetches/sec	RAM/Quota Usage	Data/Disk Usage		
gamesim-sample	● 4	586	0	0	100MB / 400MB	10.9MB / 27MB	Documents	Views
NYC-Bucket	● 4	0	0	0	98.6MB / 400MB	10.6MB / 10.6MB	Documents	Views

Next, let's configure encrypted, XDCR bidirectional replication. In this 6-node(NYC Cluster) cluster's UI, click on XDCR at the top and then click "Create Cluster Reference":



Couchbase - 6 Node NYC Cluster Enterprise Edition

Replications

REMOTE CLUSTERS — Create Cluster Reference

Name	IP/hostname

No cluster references defined. Please create one.

ONGOING REPLICATIONS

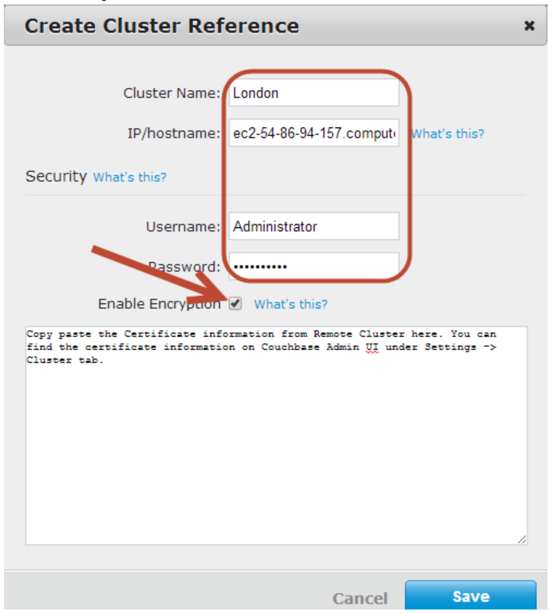
Bucket	Protocol	From	To	Filtered	Status	When

There are no replications currently in progress.

In the Create Cluster Reference popup, enter the following:

Cluster Name: London
IP/hostname: <Node #1 in the remote/London cluster's Public Hostname (Red VM)
Username: Administrator
Password: couchbase
Enable Encryption: Place a check here

Now you will see the pop-up expand and request the encryption certificate from the remote/London cluster.

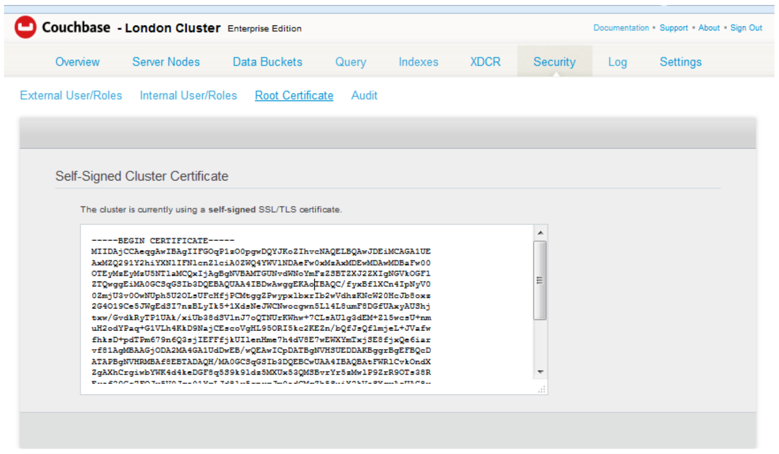


Cluster Name: London
IP/hostname: ec2-54-86-94-157.comput
Username: Administrator
Password: ••••••••••
Enable Encryption ✓

Copy paste the Certificate information from Remote Cluster here. You can find the certificate information on Couchbase Admin UI under Settings -> Cluster tab.

Quickly switch to the 2-node cluster's browser tab and click on Security(tab) at the top and then click the "Root Certificate" button.

Finally, copy the SSL certificate from the text box into your clipboard.

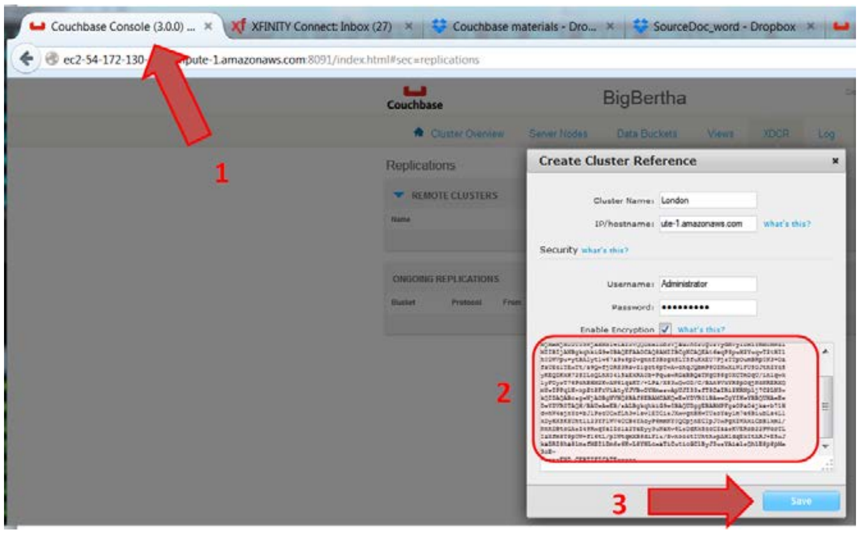


Self-Signed Cluster Certificate

The cluster is currently using a **self-signed** SSL/TLS certificate.

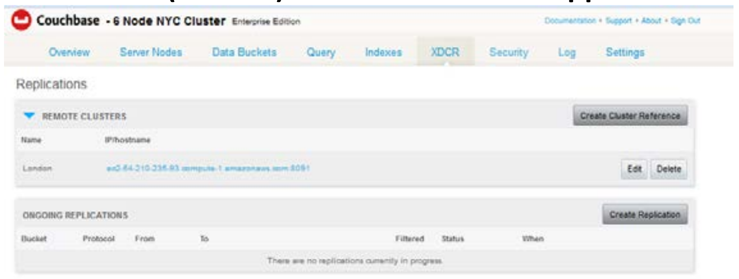
```
-----BEGIN CERTIFICATE-----
MIIDAjCCAeqgAwIBAgIIFGOqP1zO0pgwDQYJKoZIhvcNAQELBQAwJDEiMCAGA1UE
AxMZQ291Y2hiYXNlIFNlcnZlciA0ZWQ4YWV1NDAeFw0xMzAxMDEwMDAwMDBaFw00
OTEyMzEyMzU5NTlaMCQxIjAgBgNVBAMTGUNvdWNoYmFzZSBTZXJ2ZXIgNGVkOGF1
ZTQwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQC/fyxBflXCn4IpNyV0
0ZmjU3v0OwNUph5U2OLsUFcHfjPCMtggZPwypxlbxrIb2wVdhzKNcW20HcJb8oxz
2G4O19Ce5JWgEdSI7nzBLyIk5+1XdsNeJWCNwocgwn5Ll4L8umF8DGfUAxyAUShj
txw/GvdkRyTP1UAk/xiUb38dSVlnJ7oQTNUrKWhw+7CLsAUlg3dEM+Z15wcsU+nm
uH2odYPaq+G1VLh4KkD9NajCEscoVgHL95ORI5kc2KEZn/bQfJsQflmjeL+JVafw
fhksD+pdTPm679n6Q3sjIEFFfjkUIlenHme7h4dV8E7wEWXYmTxjSE8fjxQe6iar
vf81AgMBAAGjODA2MA4GA1UdDwEB/wQEAwICpDATBgNVHSUEDDAKBggrBgEFBQcD
ATAPBgNVHRMBAf8EBTADAQH/MA0GCSqGSIb3DQEBCwUAA4IBAQBAtFWR1CvkOndX
ZgAXhCrgiwbYWK4d4keDGF8q5S9k91dz5MXUx53QMSBvrYr5zMwlP9ZrR9OTs38R
```

Switch back to the other browser tab for the 6-node NYC cluster, paste the SSL certificate that you copied and click Save:



Cluster Name: London
IP/hostname: ute-1.amazonaws.com
Username: Administrator
Password: ••••••••••
Enable Encryption ✓

The remote (London) cluster should now appear under Remote Cluster:

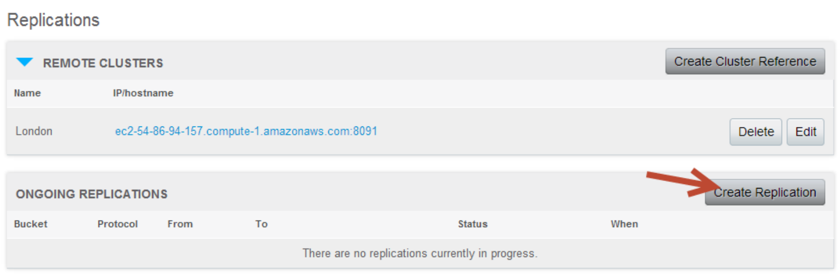


Name	IP/hostname
London	ec2-54-210-236-93.compute-1.amazonaws.com:8091

Bucket	Protocol	From	To	Filtered	Status	When

There are no replications currently in progress.

Next, let's configure XDCR bidirectional replication for the new bucket we created. In this same 6-node cluster's UI, click "Create Replication":



Name	IP/hostname
London	ec2-54-86-94-157.compute-1.amazonaws.com:8091

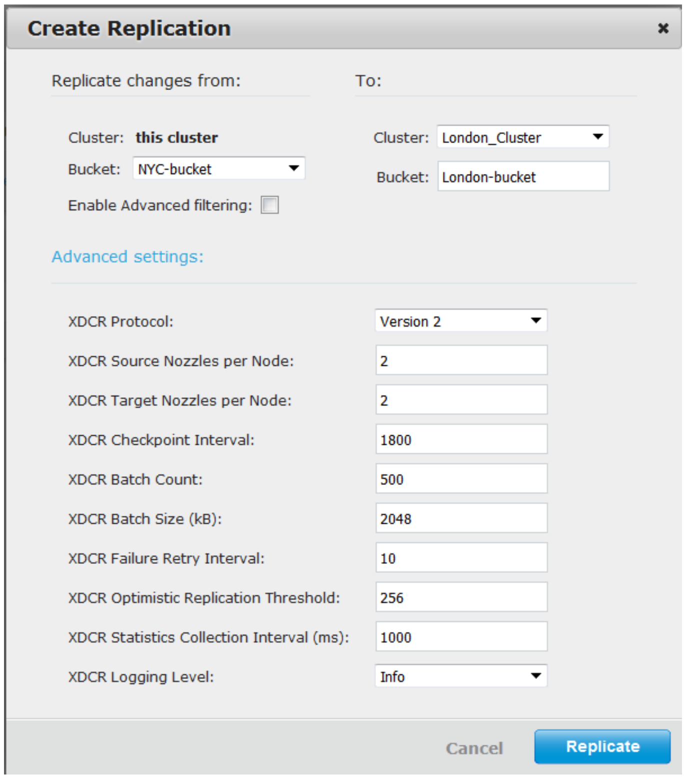
Bucket	Protocol	From	To	Status	When

There are no replications currently in progress.

On the Create Replication pop-up, for 'Replicate changes from', choose the "NYC-bucket". For the 'To Cluster', choose "London" and for the 'To' Bucket, choose "London-bucket".

Then click on Advanced settings, just to see the options, but we will not change any of them. Just notice that the Optimistic Replication Threshold is set to 256.

Click Replicate:



Create Replication

Replicate changes from: | To:

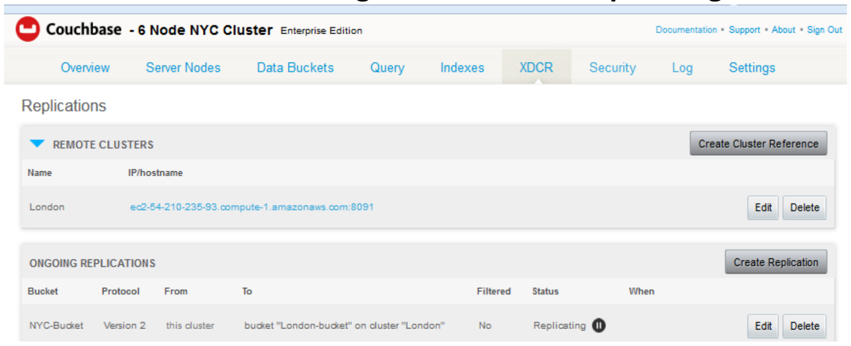
Cluster:	this cluster	Cluster:	London_Cluster
Bucket:	NYC-bucket	Bucket:	London-bucket
Enable Advanced filtering:	☐		

Advanced settings:

XDCR Protocol:	Version 2
XDCR Source Nozzles per Node:	2
XDCR Target Nozzles per Node:	2
XDCR Checkpoint Interval:	1800
XDCR Batch Count:	500
XDCR Batch Size (kB):	2048
XDCR Failure Retry Interval:	10
XDCR Optimistic Replication Threshold:	256
XDCR Statistics Collection Interval (ms):	1000
XDCR Logging Level:	Info

Cancel Replicate

The ongoing replications section will now show the replication status as "Starting Up". In a few seconds that will change to a status of Replicating:


Couchbase - 6 Node NYC Cluster Enterprise Edition
Documentation • Support • About • Sign Out
Overview
Server Nodes
Data Buckets
Query
Indexes
XDCR
Security
Log
Settings

Replications

REMOTE CLUSTERS — Create Cluster Reference

Name	IP/hostname	
London	ec2-54-210-235-93.compute-1.amazonaws.com:8091	Edit Delete

ONGOING REPLICATIONS — Create Replication

Bucket	Protocol	From	To	Filtered	Status	When	
NYC-Bucket	Version 2	this cluster	bucket "London-bucket" on cluster "London"	No	Replicating		Edit Delete

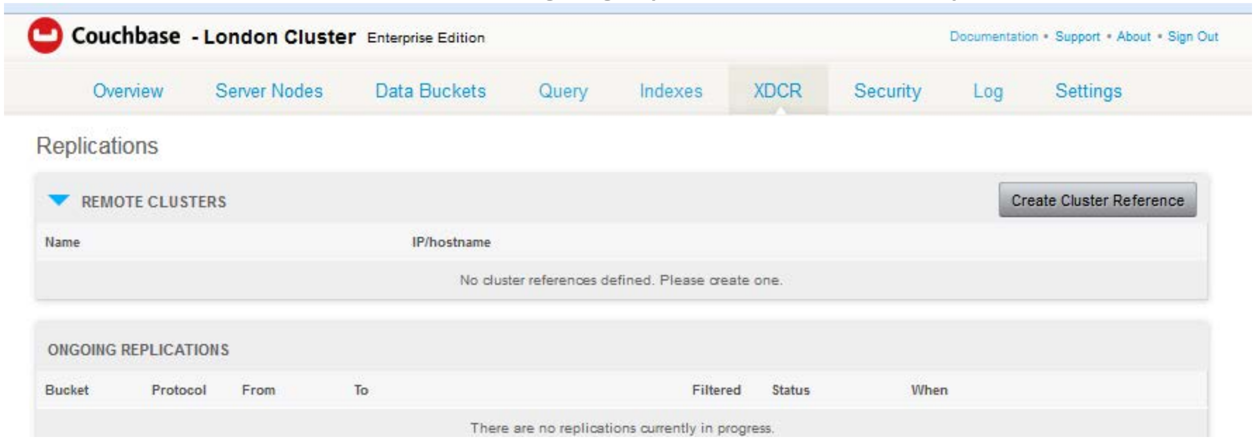
We are finished setting up the encrypted, unidirectional replication from the 6-node cluster (NYC) to the 2-node cluster (London). Let's imagine for now that the 6-node cluster is in NYC

and the 2-node cluster is in London, even though this is not the case for the actual VMs in Amazon.

Users for our global application in the United States will hit the 6-node NYC cluster for reads + writes while users in Europe will hit the 2-node London cluster for reads + writes.

Next, need to set up unidirectional replication from the 2-node London cluster back to the 6-node NYC cluster.

Switch browser tabs to the 2-node London cluster and click XDCR at the top menu. You should see no remote clusters and no ongoing replication streams set up here.



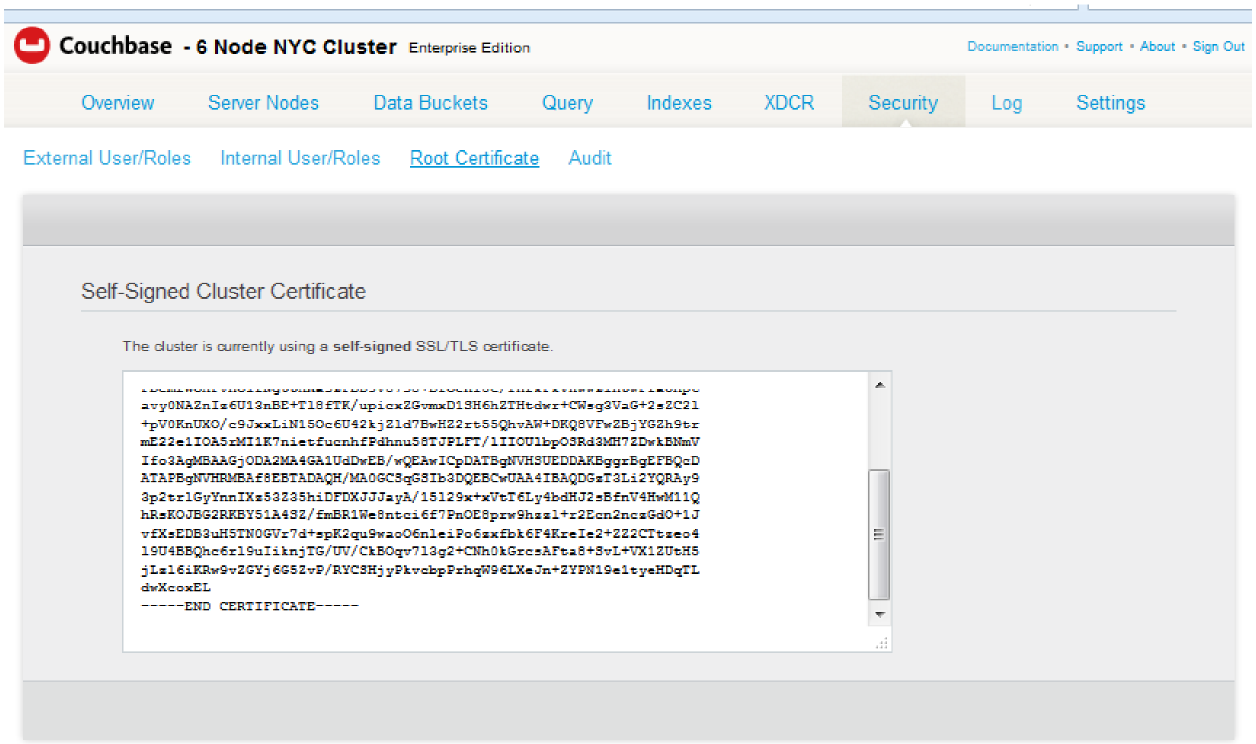
Name	IP/hostname
No cluster references defined. Please create one.	

Bucket	Protocol	From	To	Filtered	Status	When
There are no replications currently in progress.						

This time, let's use the couchbase-cli tool to create a remote XDCR cluster via an encrypted channel and then once again use the couchbase-cli tool to create a replication stream.

Note that the REST API can also be used to manage XDCR clusters and streams, but we will not explore the REST API further in this lab:
http://docs.couchbase.com/admin/admin/REST/rest-xdcr-intro.html

Switch browser tabs to the 6-node NYC cluster and click 'Security' at the top menu. Then click on "Root Certificate" , and you should see the SSL Certificate. Copy this SSL certificate to your clipboard.

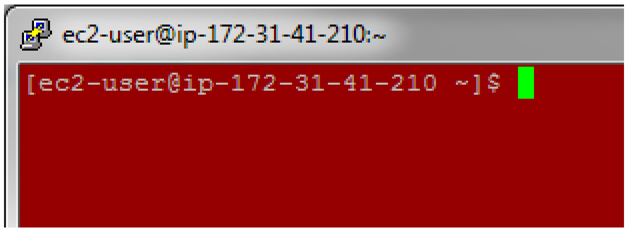


Self-Signed Cluster Certificate

The cluster is currently using a self-signed SSL/TLS certificate.

```
avy0NA2nIz6U13nBE+T18fTK/upicx2GvmxD13H6h2THtdwr+CWsg3VaG+2sZC21
+pV0KnUXO/c9JxxLiN15Oc6U42kjZ1d7BwHZ2rt55QhvAW+DKQ8VFwZBjYGZh9tr
mE22e1IOA5rMI1K7nietfucnhfPdhnu58TJPLFT/1IIOU1bpO3Rd3MH72DwkBNmV
Ifo3AgMBAAGjODA2MA4GA1UdDwEB/wQEAwICpDATBgNVHSUEDDAKBggrBgEFBQcD
ATAPBgNVHRMBAf8EBTADAQH/MA0GCSqGSIb3DQEBCwUAA4IBAQDGzT3Li2YQRAy9
3p2trlGyYnnIXz53Z35hiDFDXJJJayA/15129x+xVtT6Ly4bdHJ2sBfnV4HwM11Q
hRsKOJBG2RKBY51A43Z/fmBR1We8ntci6f7PnOE8prw9hzzl+r2Ecn2nczGdO+1J
vfXsEDB3uH5TN0GVr7d+spK2qu9waoO6nleiPo6zxfbk6F4KreIe2+ZZ2CTtzeo4
19U4BBQhc6rl9uIiknjTG/UV/CkBOqv713g2+CNh0kGrcsAFta8+3vL+VX1ZUtH5
jLzl6iKRw9vZGYj6G5ZvP/RYC3HjyPkvcbpPrhqW96LXeJn+ZYPN19e1tyeHDqTL
dwXcoxEL
-----END CERTIFICATE-----
```

Next to create a cluster reference via the command line, switch to XDCR Node #1 (Red VM) in the 2-node London cluster:

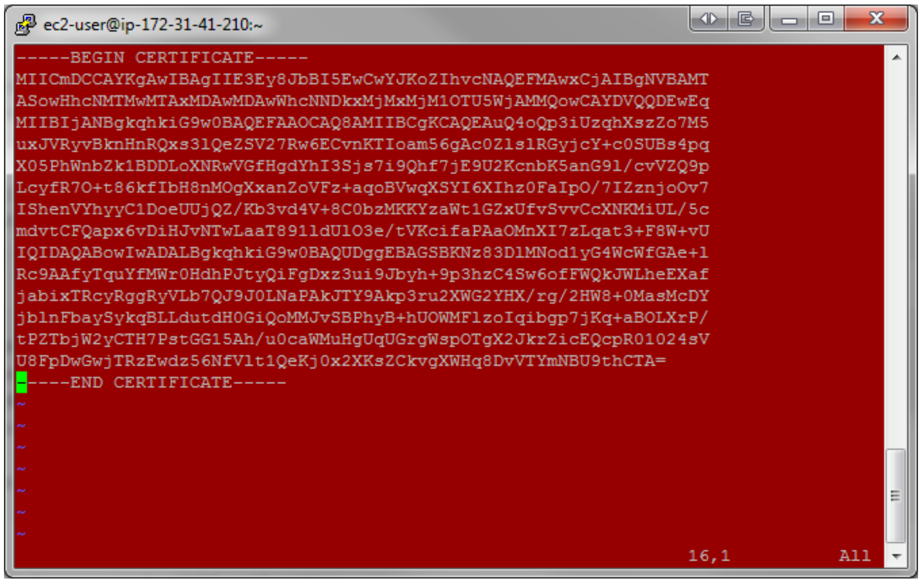


```
[ec2-user@ip-172-31-41-210 ~]$
```

Then we need to create a certificate file locally on this machine. You can use vi, vim, nano, emacs or any other linux text editor of your choice to create the .pem file. Ask the instructor for assistance if you are not familiar with any linux text editors! The vi command below may differ according to which text editor you choose.

```
[ec2-user@ip-172-31-41-210 ~]$ cd ~
[ec2-user@ip-172-31-41-210 ~]$ vi NYC-certfile.pem
```

Then paste the certificate into the text editor:



ec2-user@ip-172-31-41-210:~

```
-----BEGIN CERTIFICATE-----
MIICmDCCAYKgAwIBAgIIE3Ey8JbBI5EwCwYJKoZIhvcNAQEFMAwxCjAIBgNVBAMT
ASowHhcNMTMwMTAxMDAwMDAwWhcNNDkxMjMxMjM1OTU5WjAMMQowCAYDVQQDEwEq
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAuQ4oQp3iUzqhXszZo7M5
uxJVRyvBknHnRQxs3lQeZSV27Rw6ECvnKTIoam56gAc0ZlslRGyjcY+c0SUBs4pq
X05PhWnbZk1BDDLoXNRwVGfHgdYhI3Sjs7i9Qhf7jE9U2KcnbK5anG9l/cvVZQ9p
LcyfR7O+t86kfIbH8nMOgXxanZoVFz+aqoBVwqXSYI6XIhz0FaIpO/7IZznjoOv7
IShenVYhyyC1DoeUUjQZ/Kb3vd4V+8C0bzMKKYzaWt1GZxUfvSvvCcXNKMiUL/5c
mdvtCFQapx6vDiHJvNTwLaaT891ldUlO3e/tVKcifaPAaOMnXI7zLqat3+F8W+vU
IQIDAQABowIwADALBgkqhkiG9w0BAQUDggEBAGSBKNz83DlMNod1yG4WcWfGAe+l
Rc9AAfyTquYfMWr0HdhPJtyQiFgDxz3ui9Jbyh+9p3hzC4Sw6ofFWQkJWLheEXaf
jabixTRcyRggRyVLb7QJ9J0LNaPAkJTY9Akp3ru2XWG2YHX/rg/2HW8+0MasMcDY
jblnFbaySykqBLLdutdH0GiQoMMJvSBPhyB+hUOWMFlzoIqibgp7jKq+aBOLXrP/
tPZTbjW2yCTH7PstGG15Ah/u0caWMuHgUqUGrgWspOTgX2JkrZicEQcpR01024sV
U8FpDwGwjTRzEwdz56NfVlt1QeKj0x2XKsZCkvgXWHq8DvVTYmNBU9thCTA=
-----END CERTIFICATE-----
~
~
~
~
~
~
~
                                                  16,1          All
```

Finally save and quit out of the file.

If you cat the file, you should see the certificate:

```
[ec2-user@ip-172-31-41-210 ~]$ cat NYC-certfile.pem
-----BEGIN CERTIFICATE-----
MIICmDCCAYKgAwIBAgIIE3Ey8JbBI5EwCwYJKoZIhvcNAQEFMAwxCjAIBgNVBAMT
ASowHhcNMTMwMTAxMDAwMDAwWhcNNDkxMjMxMjM1OTU5WjAMMQowCAYDVQQDEwEq
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAuQ4oQp3iUzqhXszZo7M5
uxJVRyvBknHnRQxs3lQeZSV27Rw6ECvnKTIoam56gAc0ZlslRGyjcY+c0SUBs4pq
X05PhWnbZk1BDDLoXNRwVGfHgdYhI3Sjs7i9Qhf7jE9U2KcnbK5anG9l/cvVZQ9p
LcyfR7O+t86kfIbH8nMOgXxanZoVFz+aqoBVwqXSYI6XIhz0FaIpO/7IZznjoOv7
IShenVYhyyC1DoeUUjQZ/Kb3vd4V+8C0bzMKKYzaWt1GZxUfvSvvCcXNKMiUL/5c
mdvtCFQapx6vDiHJvNTwLaaT891ldUlO3e/tVKcifaPAaOMnXI7zLqat3+F8W+vU
IQIDAQABowIwADALBgkqhkiG9w0BAQUDggEBAGSBKNz83DlMNod1yG4WcWfGAe+l
Rc9AAfyTquYfMWr0HdhPJtyQiFgDxz3ui9Jbyh+9p3hzC4Sw6ofFWQkJWLheEXaf
jabixTRcyRggRyVLb7QJ9J0LNaPAkJTY9Akp3ru2XWG2YHX/rg/2HW8+0MasMcDY
jblnFbaySykqBLLdutdH0GiQoMMJvSBPhyB+hUOWMFlzoIqibgp7jKq+aBOLXrP/
tPZTbjW2yCTH7PstGG15Ah/u0caWMuHgUqUGrgWspOTgX2JkrZicEQcpR01024sV
U8FpDwGwjTRzEwdz56NfVlt1QeKj0x2XKsZCkvgXWHq8DvVTYmNBU9thCTA=
-----END CERTIFICATE-----
```

Finally, submit the following command to create the cluster reference. Note that the --xdcr-hostname attribute will be the public hostname for Node #1 (dark blue VM) in the 6-node cluster:

```
[ec2-user@ip-172-31-41-210 ~]$  /opt/couchbase/bin/couchbase-cli xdcr-
setup -c localhost:8091 -u Administrator -p couchbase --create --xdcr-
cluster-name=NYC --xdcr-hostname=ec2-54-85-43-128.compute-
1.amazonaws.com:8091 --xdcr-username=Administrator --xdcr-
password=couchbase --xdcr-demand-encryption=1 --xdcr-
certificate=/home/ec2-user/NYC-certfile.pem
```

```
SUCCESS: init/edit NYC
```

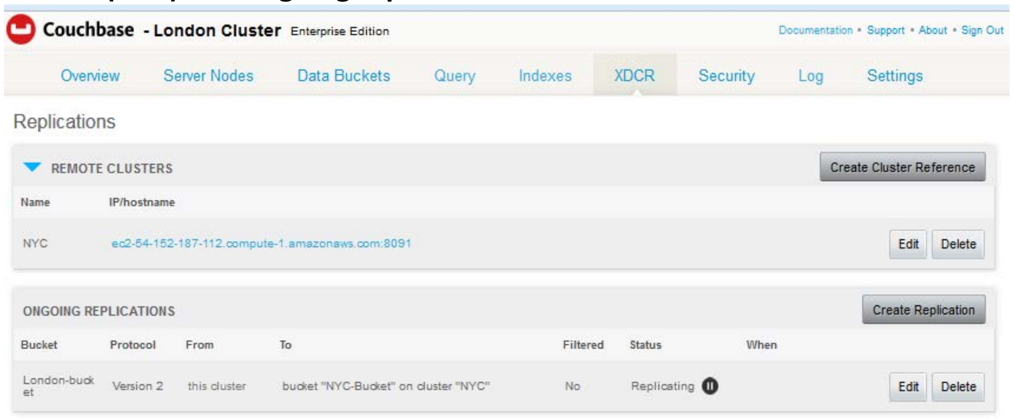
The above SUCCESS message means the cluster reference creation was successful.

Almost there. Now run the following command to start a replication stream from London-bucket back to NYC-bucket:

```
[ec2-user@ip-172-31-41-210 ~]$ /opt/couchbase/bin/couchbase-cli xdcr-
replicate -c localhost:8091 -u Administrator -p couchbase --create --
xdcr-cluster-name=NYC --xdcr-from-bucket=London-bucket --xdcr-to-
bucket=NYC-bucket

SUCCESS: start replication
```

The above SUCCESS message means the replication stream creation was successful.

Next, switch over to the 2-node London cluster's Web UI and click on XDCR to see the Remote Cluster (NYC) and ongoing replication in the GUI:



REMOTE CLUSTERS

Name	IP/hostname
NYC	ec2-54-152-187-112.compute-1.amazonaws.com:8091

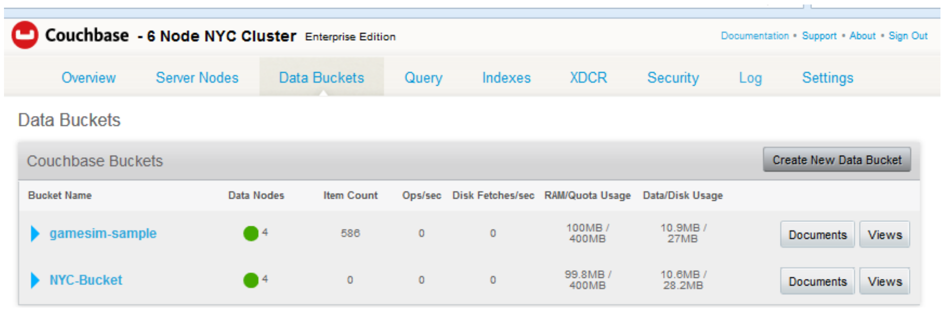
ONGOING REPLICATIONS

Bucket	Protocol	From	To	Filtered	Status	When
London-bucket	Version 2	this cluster	bucket "NYC-Bucket" on cluster "NYC"	No	Replicating	

You may see some errors in the replication status, but you can ignore these for now and continue.

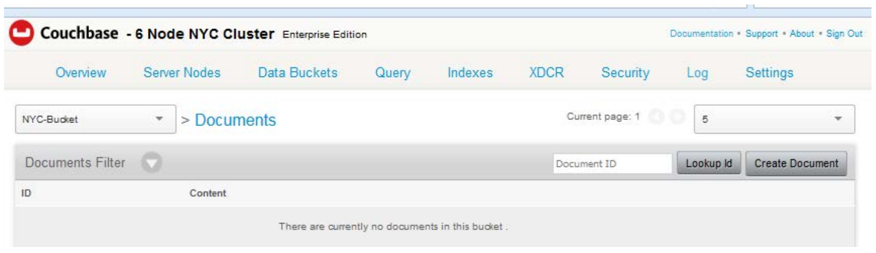
Now that our encrypted, bidirectional replication is set up, let's create a simple item in NYC/6-node cluster and see if it replicates to the London side. And then let's create a simple item in London/2-node cluster and see if it replicates to the NYC side.
We'll create the simple item in the NYC cluster first.

Switch to the 6-node NYC cluster's browser tab. Then click "Data Buckets" in the top menu and finally click on Documents:


Couchbase - 6 Node NYC Cluster Enterprise Edition
Documentation • Support • About • Sign Out
Overview Server Nodes Data Buckets Query Indexes XDCR Security Log Settings
Data Buckets
Couchbase Buckets
Create New Data Bucket
Documents Views

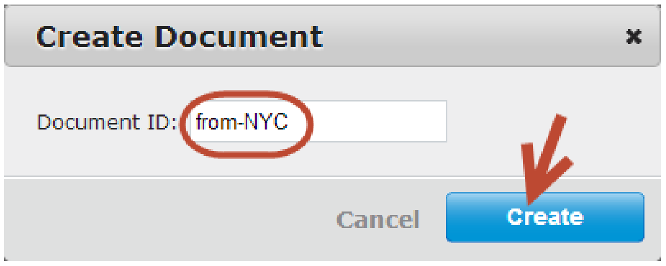
Bucket Name	Data Nodes	Item Count	Ops/sec	Disk Fetches/sec	RAM/Quota Usage	Data/Disk Usage
gamesim-sample	● 4	586	0	0	100MB / 400MB	10.9MB / 27MB
NYC-Bucket	● 4	0	0	0	99.8MB / 400MB	10.6MB / 28.2MB

Click "Create Document":

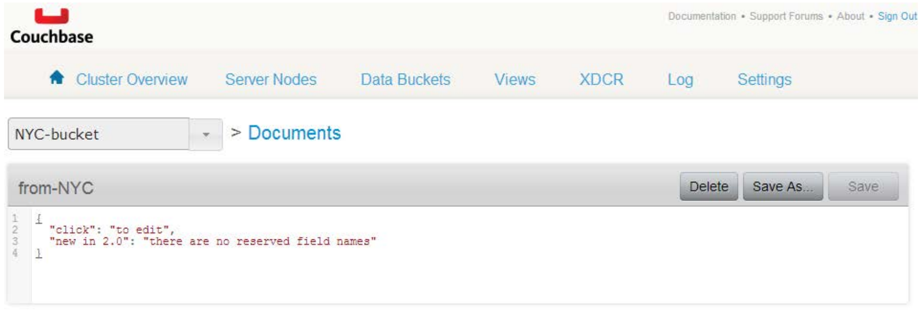

Couchbase - 6 Node NYC Cluster Enterprise Edition
Documentation • Support • About • Sign Out
Overview Server Nodes Data Buckets Query Indexes XDCR Security Log Settings
NYC-Bucket
> Documents
Current page: 1
5
Documents Filter
Document ID
Lookup Id
Create Document
ID
Content

There are currently no documents in this bucket .

Name the Document ID "from-NYC" and click Create:

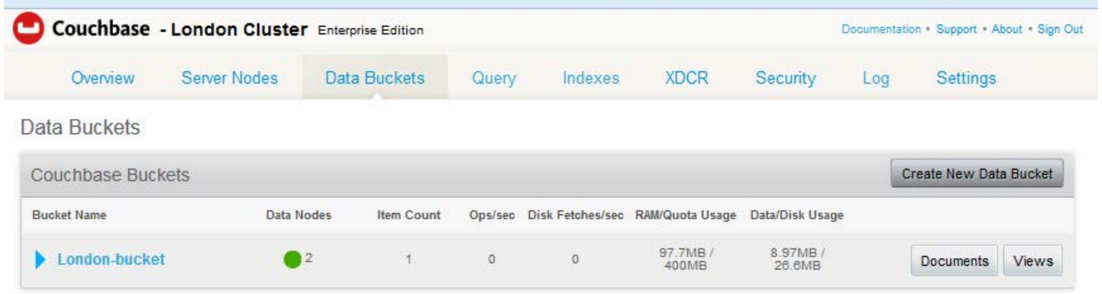


The default JSON document now appears in the UI:


Couchbase
Documentation • Support Forums • About • Sign Out
Cluster Overview Server Nodes Data Buckets Views XDCR Log Settings
NYC-bucket
> Documents
from-NYC
Delete Save As... Save

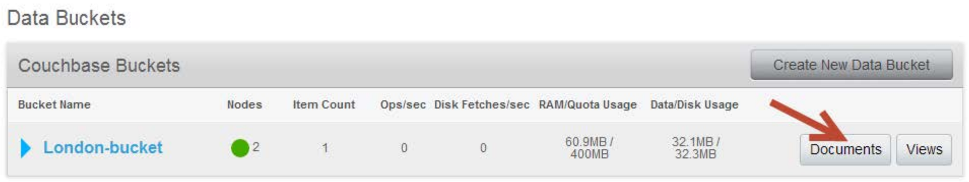
```
{
  "click": "to edit",
  "new in 2.0": "there are no reserved field names"
}
```

Switch to the 2-node London cluster's browser tab. Then click "Data Buckets" in the top menu. You may have to refresh the page before you see the 1 item count on this side.



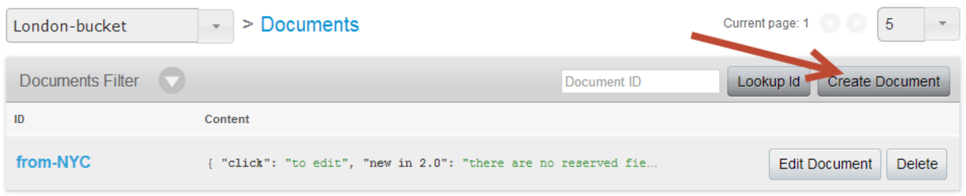
Bucket Name	Data Nodes	Item Count	Ops/sec	Disk Fetches/sec	RAM/Quota Usage	Data/Disk Usage
London-bucket	● 2	1	0	0	97.7MB / 400MB	8.97MB / 26.6MB

So, the item we created in NYC successfully replicated to London.
Click Documents:



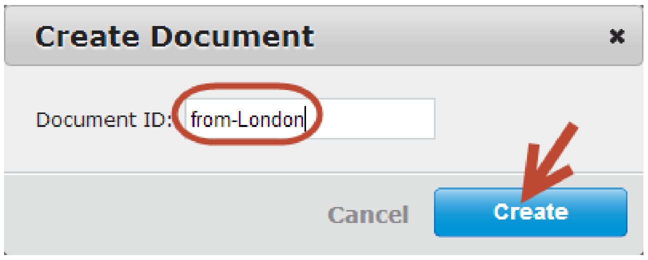
Bucket Name	Nodes	Item Count	Ops/sec	Disk Fetches/sec	RAM/Quota Usage	Data/Disk Usage
London-bucket	● 2	1	0	0	60.9MB / 400MB	32.1MB / 32.3MB

You should now see the 'from-NYC' key here in London. Click Create Document:



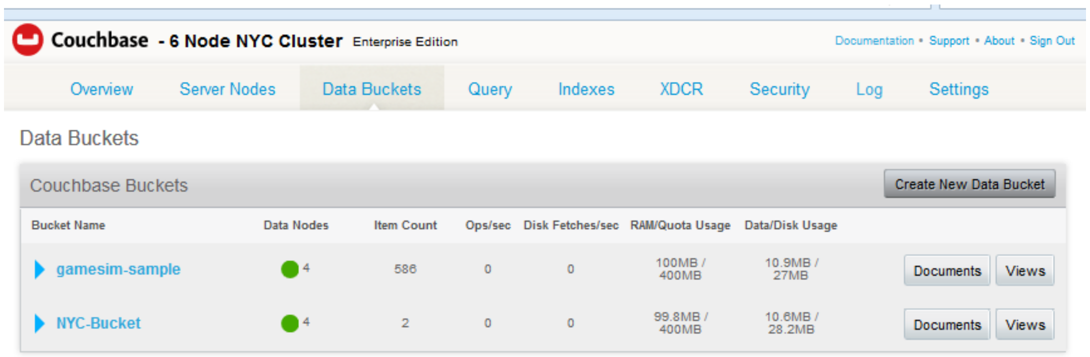
ID	Content
from-NYC	{ "click": "to edit", "new in 2.0": "there are no reserved fie…

In the pop-up, name the Document ID "from-London" and click Create:



Document ID: from-London

Now switch back to the 6-node NYC cluster's browser tab to verify that the 2nd key we created shows up there.
Click 'Data Buckets' and see that the item count is 2 (you may have to refresh the page). Next click on Documents:

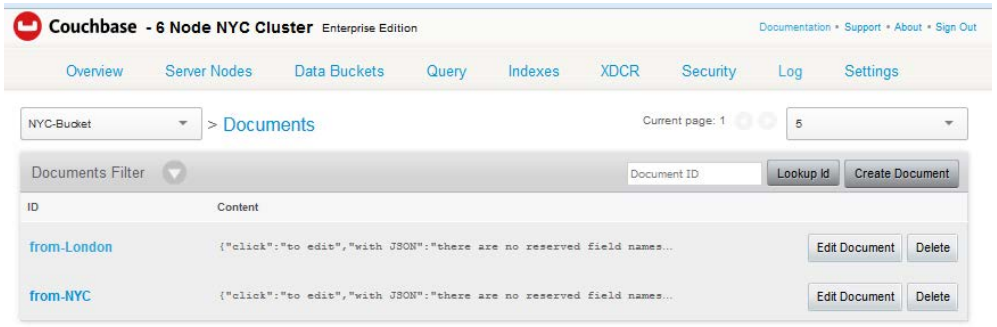


Data Buckets

Couchbase Buckets

Bucket Name	Data Nodes	Item Count	Ops/sec	Disk Fetches/sec	RAM/Quota Usage	Data/Disk Usage
gamesim-sample	● 4	586	0	0	100MB / 400MB	10.9MB / 27MB
NYC-Bucket	● 4	2	0	0	99.8MB / 400MB	10.6MB / 28.2MB

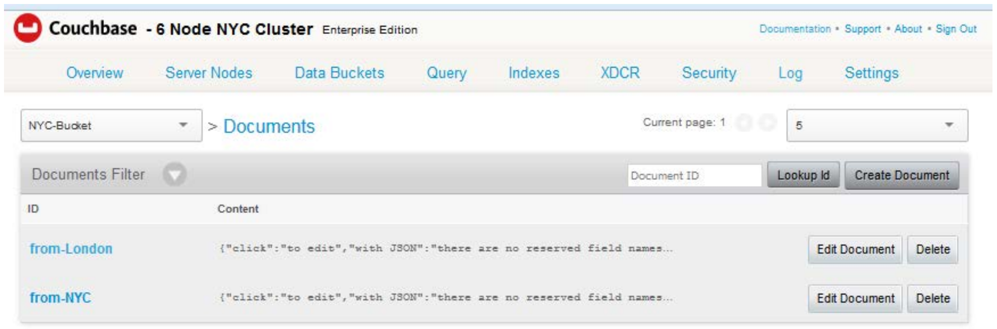
You should now see both keys in the NYC cluster. Excellent, we have now verified that bidirectional XDCR is working!



ID	Content
from-London	{"click":"to edit","with JSON":"there are no reserved field names...
from-NYC	{"click":"to edit","with JSON":"there are no reserved field names...

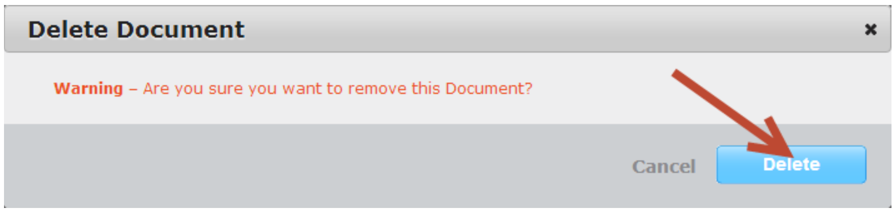
What will happen if we delete both keys from NYC? Do you think both keys will also get deleted from London? Let's find out…

Click Delete next to the 'from-London' key to delete it:

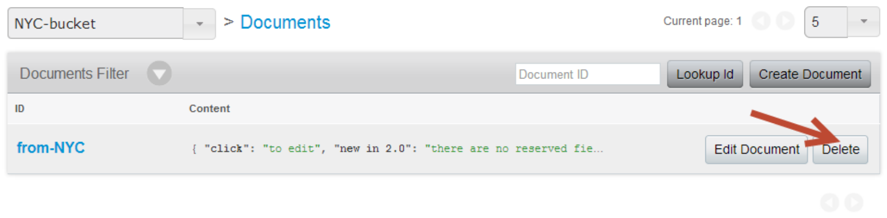


ID	Content
from-London	{"click":"to edit","with JSON":"there are no reserved field names...
from-NYC	{"click":"to edit","with JSON":"there are no reserved field names...

Confirm the delete by clicking Delete again:

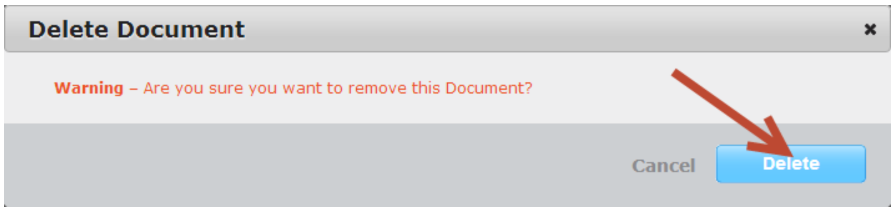


Click Delete for the 'from-NYC' key also:

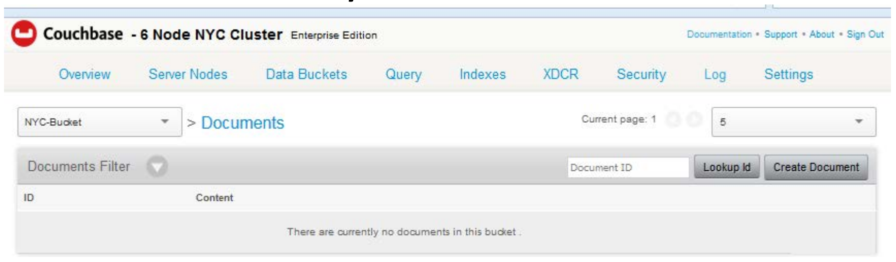


```
{ "click": "to edit", "new in 2.0": "there are no reserved fie…
```

Confirm the delete by clicking Delete again:

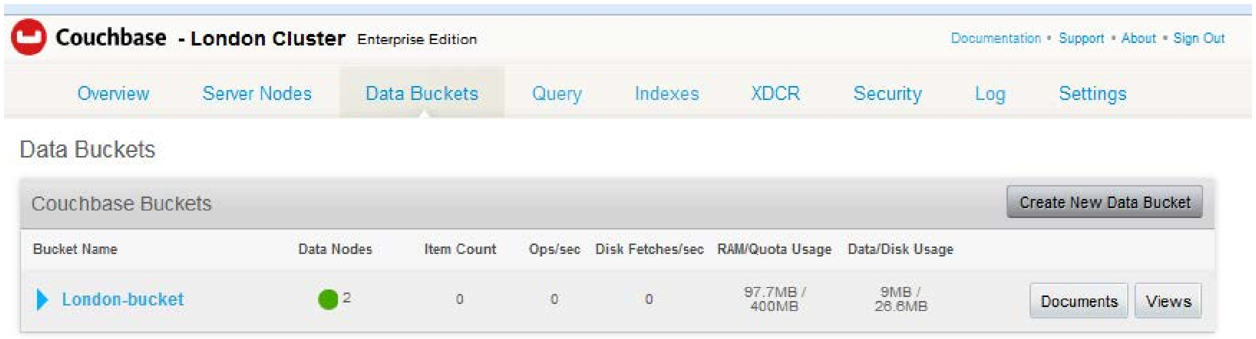


There will now be zero keys in the NYC cluster's NYC-bucket:



There are currently no documents in this bucket .

Switch to the browser tab for the 2-node London cluster. Then click "Data Buckets" in the top menu and notice that the item count is zero, so both keys did indeed get deleted. (You may have to refresh the page to reflect this)



Bucket Name	Data Nodes	Item Count	Ops/sec	Disk Fetches/sec	RAM/Quota Usage	Data/Disk Usage
London-bucket	● 2	0	0	0	97.7MB / 400MB	9MB / 26.6MB

This concludes Lab #6.