CS300 Couchbase NoSQL Server Administration

Lab 5 Exercise Manual

Couchbase
Learning Services

Release: 4.5

Revised: July 26th, 2016

Lab #5: Server Warmup, Design Docs, Views and Indexes

Objective: This 1-hour lab will first walk you through adding nodes #3 and #4 back into the cluster. Then we will explore some commands related to server warmup. Finally, the majority of the lab will walk you through creating design documents, views and indexes

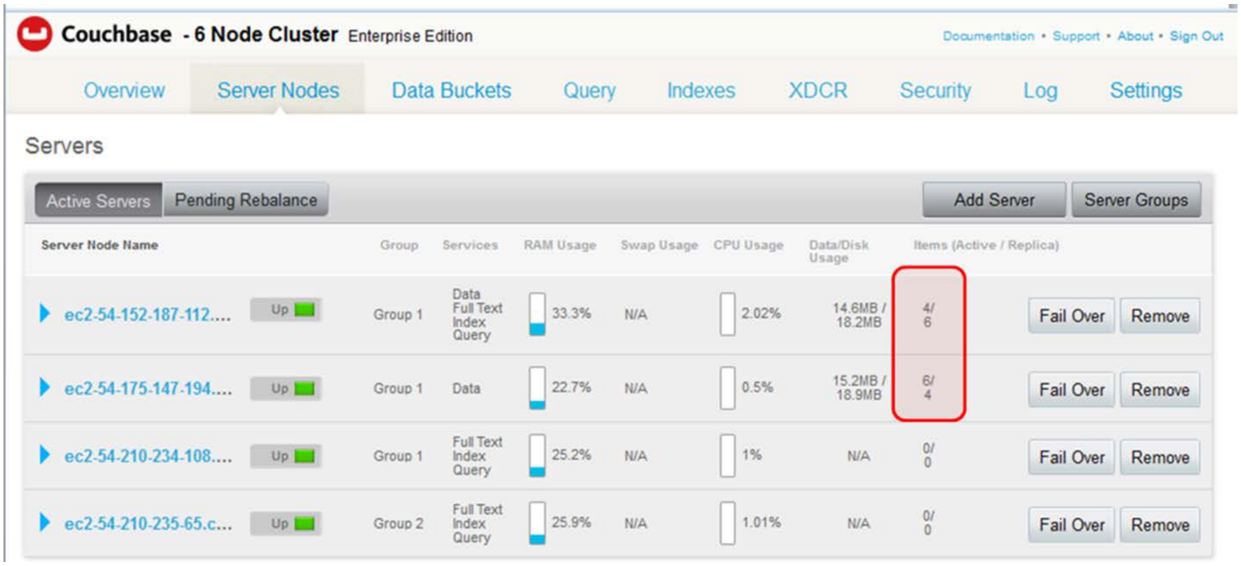
Overview: The following high-level steps are involved in this lab:

- Add nodes #3 and #4 back into cluster
- Add two sample buckets for beer-sample and gamesim-sample into the cluster
- Use cbstats command to view warmup metrics
- Locate the warmup access log file on disk
- Use the cbepctl tool to change the alog_sleep_time and flush_param settings for the access log
- Run the 'brewery-beers' view that comes by default with the beer-sample bucket
- Learn how to filter results when running a view
- Write a custom View with Map and Reduce code in development mode and then publish it to production
- View analytics and metrics for a view
- Learn how to compact the index files for a view
- Locate the view/index files on disk
- Learn how to query a view via the REST API
- Use the REST API to retrieve a design document
- Use the REST API to change the updateInterval for how often the indexer runs
- Generate load against the beer-sample bucket and watch the indexer run every 7 seconds

Add Nodes #3 and #4 back into the Cluster:

The last lab concluded with leaving the cluster in a degraded state with just 2 healthy data service nodes (Node #1 and #2) running in Server Group 1. In this section, we'll re-join Nodes #3 and #4 into the existing cluster using the Web UI.

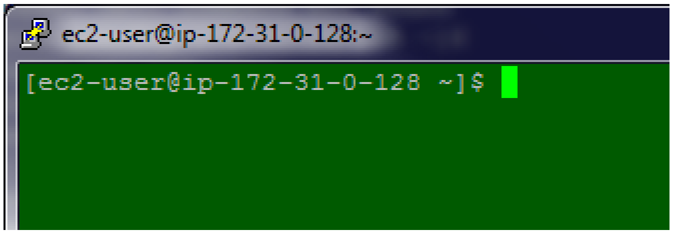
The Server Nodes page should currently show 2 nodes in your environment, which 5 Active items on each of the 2 nodes. In your specific environment, you may see 6 items on one node and 4 items on the other node… this is normal also. The items in Couchbase will not necessarily be split exactly evenly across the 2 nodes every time.



Server Node Name		Group	Services	RAM Usage	Swap Usage	CPU Usage	Data/Disk Usage	Items (Active / Replica)	
ec2-54-152-187-112....	Up 🟩	Group 1	Data Full Text Index Query	33.3%	N/A	2.02%	14.6MB / 18.2MB	4/ 6	Fail Over Remove
ec2-54-175-147-194....	Up 🟩	Group 1	Data	22.7%	N/A	0.5%	15.2MB / 18.9MB	6/ 4	Fail Over Remove
ec2-54-210-234-108....	Up 🟩	Group 1	Full Text Index Query	25.2%	N/A	1%	N/A	0/ 0	Fail Over Remove
ec2-54-210-235-65.c...	Up 🟩	Group 2	Full Text Index Query	25.9%	N/A	1.01%	N/A	0/ 0	Fail Over Remove

First, we need to start the Couchbase service on Nodes #3 and #4.

Log into Node #3 (Green VM):



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

Run the following command to start Couchbase server and check its status:

```
[ec2-user@ip-172-31-0-128 ~]$ sudo /etc/init.d/couchbase-server start
Starting couchbase-server (via systemctl):                [  OK  ]
[ec2-user@Couchbase03 ~]$
```

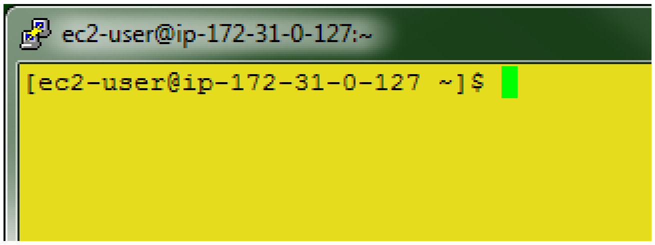
Wait 15 seconds for the service to start and then verify the status:

```
[ec2-user@ip-172-31-0-128 ~]$ sudo /etc/init.d/couchbase-server status
couchbase-server is running
```

Note you could run the following command also:

sudo systemctl start couchbase-server

Next, log into Node #4 (Yellow VM) and start Couchbase there as well:



```
[ec2-user@ip-172-31-0-127 ~]$
```

Run the following command to start Couchbase server and check its status:

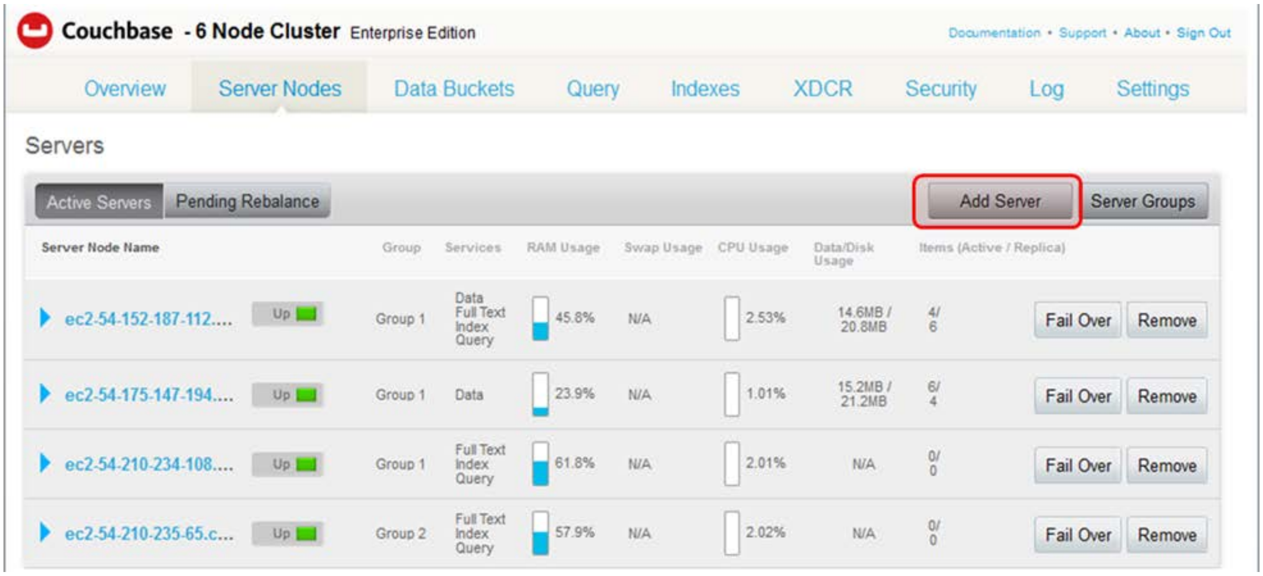
```
[ec2-user@ip-172-31-0-127~]$ sudo systemctl start couchbase-server
Starting couchbase-server                                 [  OK  ]
```

Wait 15 seconds for the service to start and then verify the status:

```
[ec2-user@ip-172-31-0-127 ~]$ sudo systemctl status couchbase-server
● couchbase-server.service - Couchbase Server
   Loaded: loaded (/usr/lib/systemd/system/couchbase-server.service; enabled; vendor preset: 
disabled)
   Active: active (running) since Wed 2016-07-20 14:59:05 EDT; 22s ago
     Docs: http://docs.couchbase.com
  Process: 29766 ExecStop=/opt/couchbase/bin/couchbase-server -k (code=exited, status=0/SUCCESS)
  Process: 2061 ExecStart=/opt/couchbase/bin/couchbase-server -- -noinput -detached (code=exited, 
status=0/SUCCESS)
 Main PID: 2131 (beam.smp)
   CGroup: /system.slice/couchbase-server.service
           ├─2074 /opt/couchbase/lib/erlang/erts-5.10.4.0.0.1/bin/epmd -daemo...
           ├─2106 /opt/couchbase/lib/erlang/erts-5.10.4.0.0.1/bin/beam.smp -A...
           ├─2131 /opt/couchbase/lib/erlang/erts-5.10.4.0.0.1/bin/beam.smp -A...
           ├─2159 sh -s disksup
           ├─2161 /opt/couchbase/lib/erlang/lib/os_mon-2.2.14/priv/bin/cpu_su...
           ├─2162 /opt/couchbase/lib/erlang/lib/os_mon-2.2.14/priv/bin/memsup...
           ├─2163 inet_gethost 4
           ├─2164 inet_gethost 4
           ├─2292 /opt/couchbase/lib/erlang/erts-5.10.4.0.0.1/bin/beam.smp -P...
           ├─2317 sh -s disksup
           ├─2318 /opt/couchbase/lib/erlang/lib/os_mon-2.2.14/priv/bin/memsup...
           ├─2319 /opt/couchbase/lib/erlang/lib/os_mon-2.2.14/priv/bin/cpu_su...
           ├─2324 /opt/couchbase/bin/priv/godu
           ├─2325 sh -s ns_disksup
           ├─2329 /opt/couchbase/bin/priv/godu
           ├─2334 /opt/couchbase/bin/saslauthd-port
           ├─2336 portsigar for ns_1@ec2-54-209-38-55.compute-1.amazonaws.com...
           ├─2366 /opt/couchbase/bin/goport
           ├─2370 /opt/couchbase/bin/goxdcr -localProxyPort=11215 -sourceKVAd...
           ├─2379 /opt/couchbase/lib/erlang/erts-5.10.4.0.0.1/bin/beam.smp -P...
           └─2396 /opt/couchbase/bin/memcached -C /opt/couchbase/var/lib/couc...

Jul 20 14:59:01 Couchbase04 systemd[1]: Starting Couchbase Server...
Jul 20 14:59:02 Couchbase04 systemd[1]: PID file /opt/couchbase/var/lib/couc....
Jul 20 14:59:05 Couchbase04 systemd[1]: couchbase-server.service: Supervisin....
Jul 20 14:59:05 Couchbase04 systemd[1]: Started Couchbase Server.
Hint: Some lines were ellipsized, use -l to show in full.
```

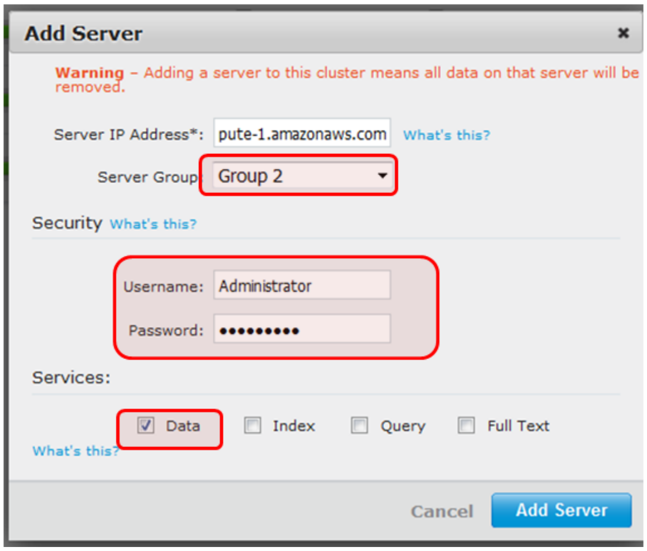
Switch back to the Web UI and under the Server Nodes page, click on Add Server:



Server Node Name		Group	Services	RAM Usage	Swap Usage	CPU Usage	Data/Disk Usage	Items (Active / Replica)		
ec2-54-152-187-112....	Up	Group 1	Data Full Text Index Query	45.8%	N/A	2.53%	14.6MB / 20.8MB	4/ 6	Fail Over	Remove
ec2-54-175-147-194....	Up	Group 1	Data	23.9%	N/A	1.01%	15.2MB / 21.2MB	6/ 4	Fail Over	Remove
ec2-54-210-234-108....	Up	Group 1	Full Text Index Query	61.8%	N/A	2.01%	N/A	0/ 0	Fail Over	Remove
ec2-54-210-235-65.c...	Up	Group 2	Full Text Index Query	57.9%	N/A	2.02%	N/A	0/ 0	Fail Over	Remove

In the popup, enter the public hostname of Node #3, choose Server Group "Group 2", use "Administrator" for the username and "couchbase" for the password. Then click "Add Server":

Add servers back into the cluster as data service nodes



Add Server ×

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

Server IP Address*: pute-1.amazonaws.com What's this?

Server Group: Group 2

Security What's this?

Username: Administrator

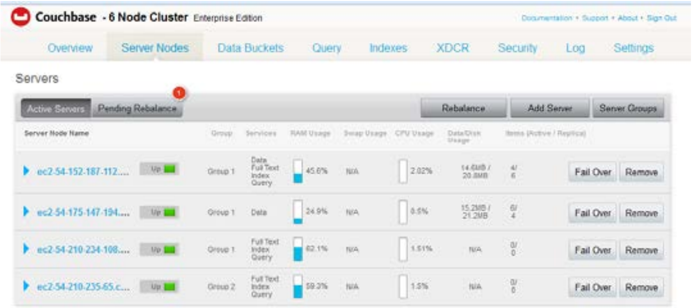
Password: •••••••••

Services:

☑ Data □ Index □ Query □ Full Text

What's this?

You should now see a red #1 under Pending Rebalance. Ignore that and click on Add Server again to add the 4th node:

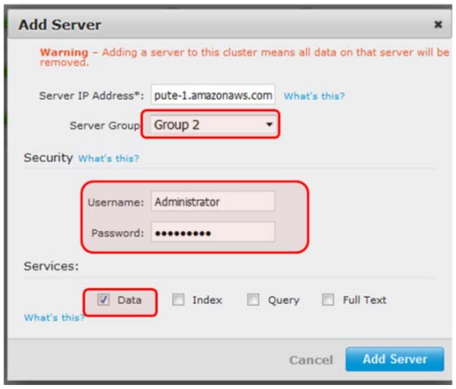


Couchbase - 6 Node Cluster Enterprise Edition

Servers

Server Node Name		Group	Services	RAM Usage	Swap Usage	CPU Usage	Data/Disk Usage	Items (Active / Replica)		
ec2-54-152-187-112....	Up	Group 1	Data Full Text Index Query	45.6%	N/A	2.02%	14.6MB / 20.8MB	4/ 6	Fail Over	Remove
ec2-54-175-147-194....	Up	Group 1	Data	24.9%	N/A	0.5%	15.2MB / 21.2MB	6/ 4	Fail Over	Remove
ec2-54-210-234-108....	Up	Group 1	Full Text Index Query	62.1%	N/A	1.51%	N/A	0/ 0	Fail Over	Remove
ec2-54-210-235-65.c...	Up	Group 2	Full Text Index Query	59.3%	N/A	1.5%	N/A	0/ 0	Fail Over	Remove

Enter the 4th node's public hostname, choose Group 2, and enter 'couchbase' for the password and click Add Server:



Add Server ×

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

Server IP Address*: pute-1.amazonaws.com What's this?

Server Group: Group 2

Security What's this?

Username: Administrator

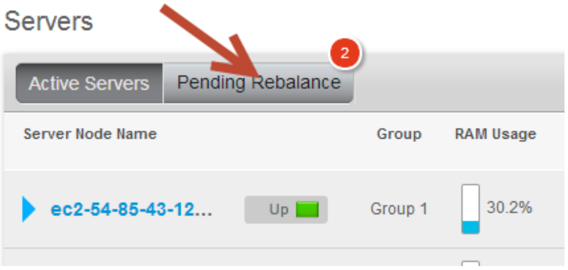
Password: •••••••••

Services:

☑ Data □ Index □ Query □ Full Text

What's this?

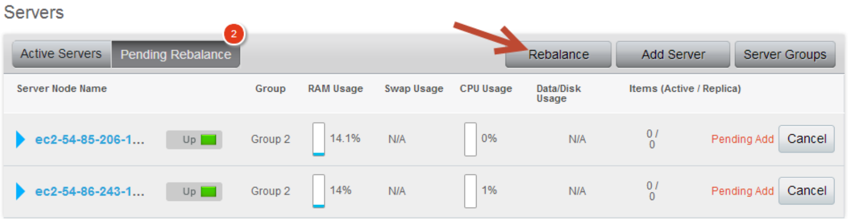
You should now see a #2 next to Pending Rebalance. Click on Pending Rebalance:



Servers

Server Node Name		Group	RAM Usage
ec2-54-85-43-12…	Up 🟩	Group 1	30.2%

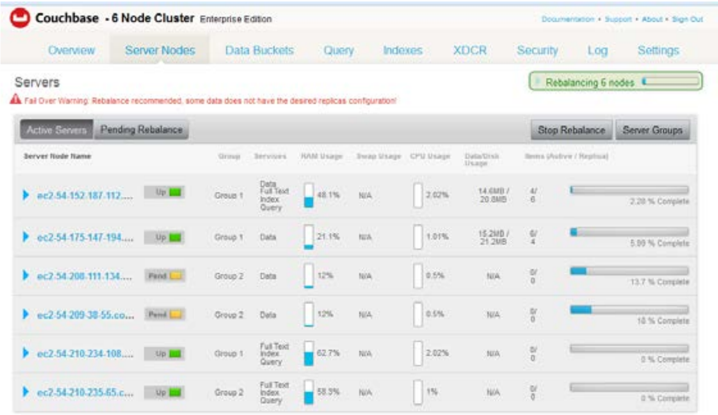
Click on Rebalance to rebalance the cluster:



Servers

Server Node Name		Group	RAM Usage	Swap Usage	CPU Usage	Data/Disk Usage	Items (Active / Replica)		
ec2-54-85-206-1…	Up 🟩	Group 2	14.1%	N/A	0%	N/A	0 / 0	Pending Add	Cancel
ec2-54-86-243-1…	Up 🟩	Group 2	14%	N/A	1%	N/A	0 / 0	Pending Add	Cancel

The rebalance operation will start running:



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

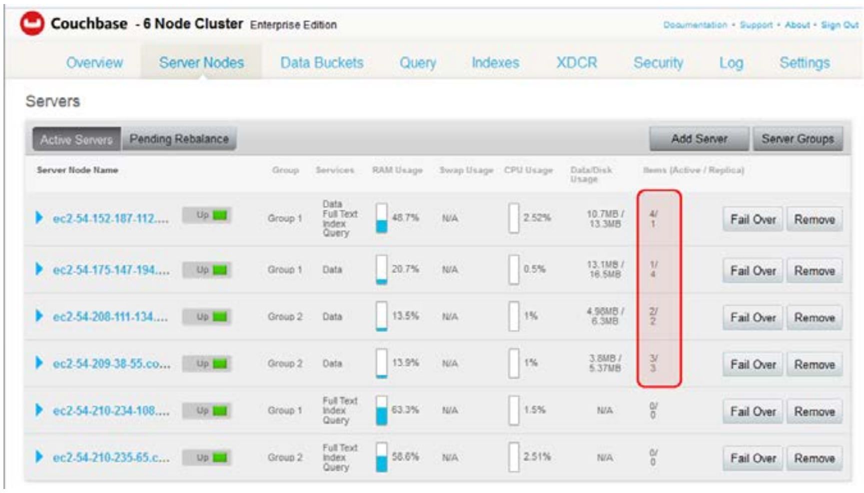
Servers

Rebalancing 6 nodes

Fail Over Warning: Rebalance recommended, some data does not have the desired replicas configuration!

Server Node Name		Group	Services	RAM Usage	Swap Usage	CPU Usage	Data/Disk Usage	Items (Active / Replica)	
ec2-54-152-187-112.…	Up 🟩	Group 1	Data Full Text Index Query	48.1%	N/A	2.02%	14.6MB / 20.8MB	4/ 6	2.28 % Complete
ec2-54-175-147-194.…	Up 🟩	Group 1	Data	21.1%	N/A	1.01%	15.2MB / 21.2MB	6/ 4	5.99 % Complete
ec2-54-208-111-134.…	Pend 🟨	Group 2	Data	12%	N/A	0.5%	N/A	0/ 0	13.7 % Complete
ec2-54-209-38-55.co…	Pend 🟨	Group 2	Data	12%	N/A	0.5%	N/A	0/ 0	18 % Complete
ec2-54-210-234-108.…	Up 🟩	Group 1	Full Text Index Query	62.7%	N/A	2.02%	N/A	0/ 0	0 % Complete
ec2-54-210-235-65.c…	Up 🟩	Group 2	Full Text Index Query	58.3%	N/A	1%	N/A	0/ 0	0 % Complete

Within about 2 minutes the cluster should fully rebalance and you should see the Items' active and replica copies scattered across the 4 data service nodes:



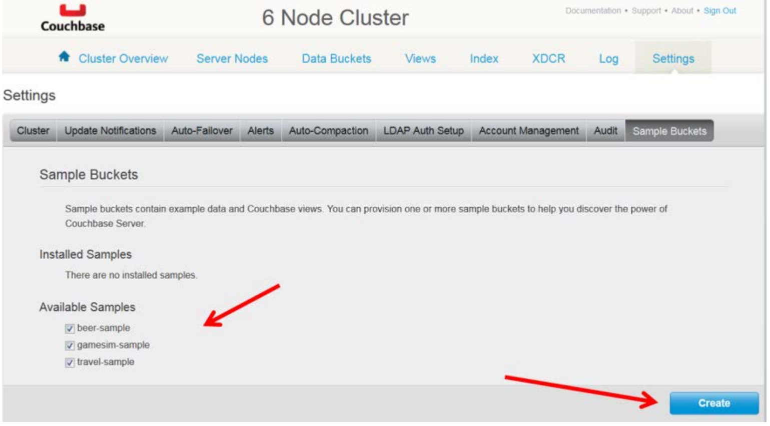
Servers

Server Node Name		Group	Services	RAM Usage	Swap Usage	CPU Usage	Data/Disk Usage	Items (Active / Replica)		
ec2-54-152-187-112....	Up	Group 1	Data Full Text Index Query	48.7%	N/A	2.52%	10.7MB / 13.3MB	4/ 1	Fail Over	Remove
ec2-54-175-147-194....	Up	Group 1	Data	20.7%	N/A	0.5%	13.1MB / 16.5MB	1/ 4	Fail Over	Remove
ec2-54-208-111-134....	Up	Group 2	Data	13.5%	N/A	1%	4.98MB / 6.3MB	2/ 2	Fail Over	Remove
ec2-54-209-38-55.co...	Up	Group 2	Data	13.9%	N/A	1%	3.8MB / 5.37MB	3/ 3	Fail Over	Remove
ec2-54-210-234-108....	Up	Group 1	Full Text Index Query	63.3%	N/A	1.5%	N/A	0/ 0	Fail Over	Remove
ec2-54-210-235-65.c...	Up	Group 2	Full Text Index Query	58.6%	N/A	2.51%	N/A	0/ 0	Fail Over	Remove

Add 3 sample buckets into the cluster:

Next we will add the beer-sample and gamesim-sample buckets back into the cluster so we can generate views upon them.

Click on Settings at the top right, then choose “Sample Buckets”. Under “Available Samples” place a check mark next to “beer-sample” , “gamesim-sample” and “travel sample” then click ‘Create’:



Settings

Sample Buckets

Sample buckets contain example data and Couchbase views. You can provision one or more sample buckets to help you discover the power of Couchbase Server.

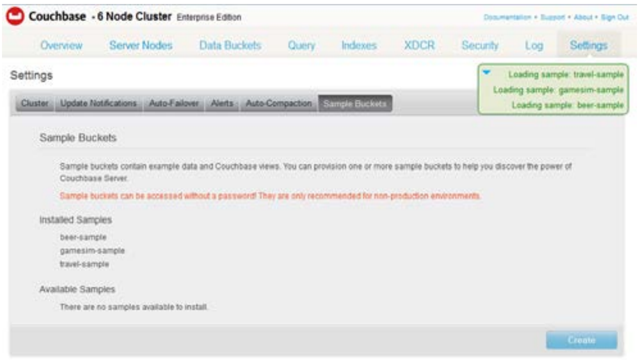
Installed Samples

There are no installed samples.

Available Samples

☑ beer-sample
☑ gamesim-sample
☑ travel-sample

When you see the “Loading sample” hover message in green in the top right corner, click on the blue arrow to expand the message:



Settings

Loading sample: travel-sample
Loading sample: gamesim-sample
Loading sample: beer-sample

Sample Buckets

Sample buckets contain example data and Couchbase views. You can provision one or more sample buckets to help you discover the power of Couchbase Server.

Sample buckets can be accessed without a password! They are only recommended for non-production environments.

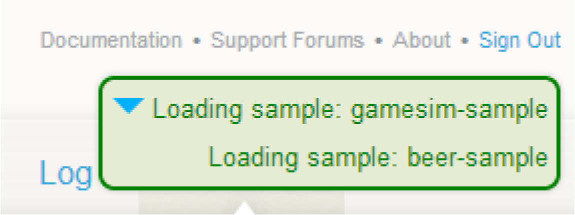
Installed Samples

beer-sample
gamesim-sample
travel-sample

Available Samples

There are no samples available to install.

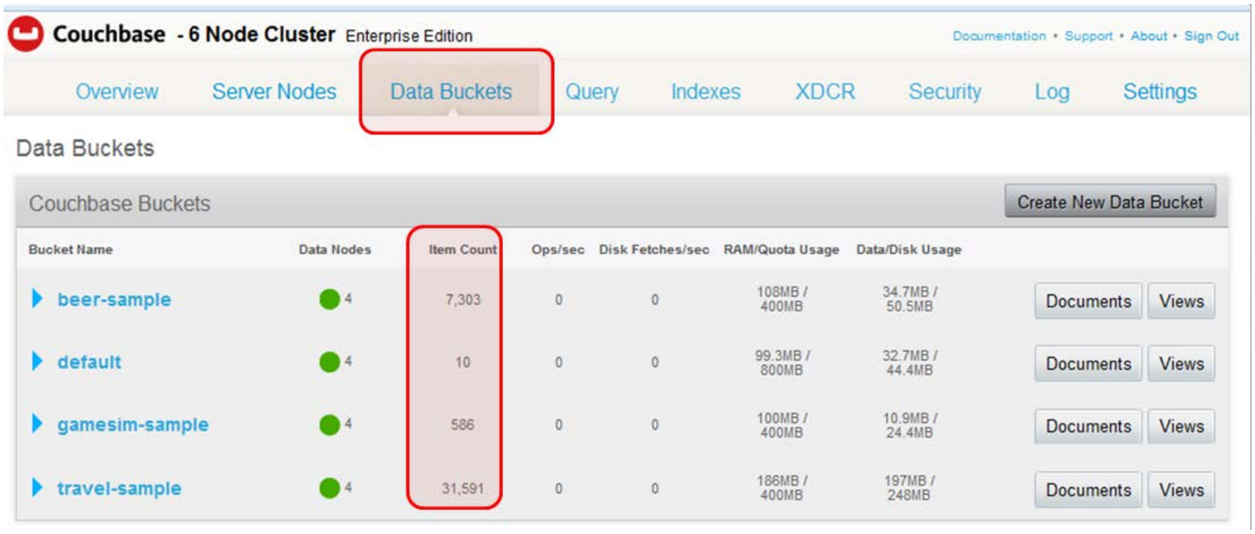
You will now see that gamesim-sample and beer-sample are both being simultaneously loaded:



Loading sample: gamesim-sample
Loading sample: beer-sample

The green hover box may also occasionally display messages about indexes being generated under the buckets.

Once the green hover box disappears, click on “Data Buckets” in the top menu to verify that the 3 sample buckets have indeed been loaded. You should see 7,303 items in the beer-sample bucket and 586 items in the gamesim-sample bucket and finally 31591 items in the travel-sample bucket:



Data Buckets

Couchbase Buckets

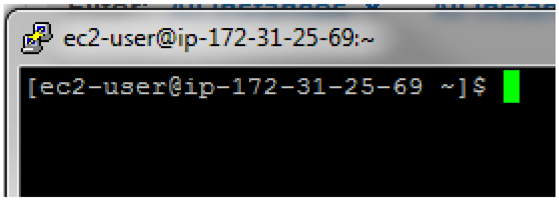
Bucket Name	Data Nodes	Item Count	Ops/sec	Disk Fetches/sec	RAM/Quota Usage	Data/Disk Usage		
beer-sample	● 4	7,303	0	0	108MB / 400MB	34.7MB / 50.5MB	Documents	Views
default	● 4	10	0	0	99.3MB / 800MB	32.7MB / 44.4MB	Documents	Views
gamesim-sample	● 4	586	0	0	100MB / 400MB	10.9MB / 24.4MB	Documents	Views
travel-sample	● 4	31,591	0	0	186MB / 400MB	197MB / 248MB	Documents	Views

Studying Server Warmup:

Anytime you restart the Couchbase Server the server must undergo a warmup process before it can handle requests for the data. During warmup the server loads data from disk into RAM; after the warmup process completes, the data is available for clients to read and write.

Depending on the size and configuration of your system and the amount of data persisted in your system, server warmup may take some time to load all of the data into memory. Couchbase Server can also be configured to begin serving data before it has actually loaded all the keys and data from vBuckets.

Switch over to the App Server (Black App Client) for the next commands:



ec2-user@ip-172-31-25-69:~

```
[ec2-user@ip-172-31-25-69 ~]$
```

Run the following cbstats command against the 3rd node's public hostname (Green VM) to see some warmup statistics for the default:

```
[ec2-user@ip-172-31-23-211 ~]$ cbstats ec2-54-86-243-136.compute-1.amazonaws.com:11210 -b default all | grep 'warmup'
 ep_failpartialwarmup:                            false
 ep_waitforwarmup:                                false
 ep_warmup:                                       true
 ep_warmup_batch_size:                            1000
 ep_warmup_dups:                                  0
 ep_warmup_min_items_threshold:                   100
 ep_warmup_min_memory_threshold:                  100
 ep_warmup_oom:                                   0
 ep_warmup_thread:                                complete
 ep_warmup_time:                                  203
```

Notice that the warmup thread for this default bucket has completed running on the 3rd node and the time it took was 203 microseconds.

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

Try running the following modified cbstats command against the 3rd node's beer-sample bucket:

```
[ec2-user@ip-172-31-23-211 ~]$ cbstats ec2-54-86-243-136.compute-1.amazonaws.com:11210 -b beer-sample all | egrep "warm|curr_items"
 curr_items:                                  1782
 curr_items_tot:                              3611
 ep_failpartialwarmup:                        false
 ep_waitforwarmup:                            false
 ep_warmup:                                   true
 ep_warmup_batch_size:                        1000
 ep_warmup_dups:                              0
 ep_warmup_min_items_threshold:               100
 ep_warmup_min_memory_threshold:              100
 ep_warmup_oom:                               0
 ep_warmup_thread:                            complete
 ep_warmup_time:                              187
 vb_active_curr_items:                        1782
 vb_pending_curr_items:                       0
 vb_replica_curr_items:                       1829
```

Here is an explanation of what some of the above metrics mean:

curr_items	The number of items currently active on this node. During warmup, this will be 0 until complete
curr_items_tot	The total number of items this node knows about (active and replica). During warmup, this will be increasing and should match ep_warmup
ep_warmup	The number of items retrieved from disk. During warmup, this should be increasing.
ep_warmup_dups	The number of duplicate items found on disk. Ideally should be 0, but a few is not a problem
ep_warmup_oom	How many times the warmup process received an Out of Memory response from the server while loading data into RAM
ep_warmup_thread	The status of the warmup thread. Can be either running or complete
ep_warmup_time:	How long the warmup thread was running for. During warmup this number should be increasing, when complete it will tell you how long the process took

Couchbase Server is able to serve information from RAM after one of the following conditions is met during warmup:

- The server has finished loading documents for all keys listed in the access log, or
- The server has finished loading documents for every key stored on disk for all vBuckets, or
- The percentage of documents loaded into memory is greater than, or equal to, the setting for `ep_warmup_min_items_threshold`, or
- If total % of RAM filled by documents is greater than, or equal to, the setting for `ep_warmup_min_memory_threshold`, or
- If total RAM usage by a node is greater than or equal to the setting for `mem_low_wat`.

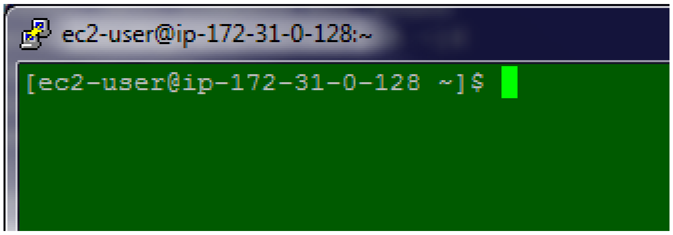
When the server reaches one of these states, this is known as the run level ; when Couchbase Server reaches this point, it immediately stops loading documents for the remaining keys. After this point, Couchbase Server will load this remaining documents from disk into RAM as a background data fetch.

Notice that the `ep_warmup_min_items_threshold` setting is currently set to 100 for this bucket. This setting indicates the percentage of documents that must be loaded into memory before warmup is considered complete. However, this does not necessarily mean that 100% of the documents definitely will get loaded in RAM before warm is complete since one of the other conditions above might be met and warmup will therefore exit.

Let's find the access log on Node #3 that is used to determine which keys were recently accessed. The server has a process that will periodically scan every key in RAM and compile them into a log, named access.log as well as maintain a backup of this access log, named access.old. The server can use this backup file during warmup if the most recent access log has been corrupted during warmup or node failure. By default this process runs initially at 2:00 GMT and will run again in 24- hour time periods after that point. You can configure this process to run at a different initial time and at a different fixed interval.

Details on how to change the access.log schedule will be introduced later in this lab.

Log into Node #3 (Green VM) to find the access log on this machine:



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

First switch to root user:

```
[ec2-user@ip-172-31-0-128 ~]$ sudo –s
```

Search for the access.log on this server:

```
[root@ip-172-31-0-128 ec2-user]# find /opt/couchbase -name access.log*
```

<no results>

Exit root:

```
[root@ip-172-31-0-128 default]# exit
exit
```

You should see no results found. This is because there is no access log created for any of the buckets yet.

Run the cbepctl command to change the default behavior of the access scanner from running every 24 hours at 2:00 am GMT to every 1 minute. Note that this is a per-node, per-bucket setting. So, if you want to change this setting for an entire Couchbase cluster, you will need to perform this command on per-node and per-bucket in the cluster. This means that if you have a data bucket that is shared by two nodes, you will nonetheless need to issue this command twice and provide the different host names and ports for each node and the bucket name.

First, check what settings can be altered with cbepctl by running it with no options:

```
[ec2-user@ip-172-31-0-128 default]# /opt/couchbase/bin/cbepctl
Usage: cbepctl host:dataport command [options]
       Note that the default dataport is 11210

Options:
 -a     iterate over all buckets (requires admin u/p)
 -p     the password for the bucket if one exists
 -b     the bucket to get stats from (Default: default)

Usage: cbepctl host:dataport drain
  or   cbepctl host:dataport set type param value
  or   cbepctl host:dataport start
  or   cbepctl host:dataport stop
Persistence:
  stop           - stop persistence
  start          - start persistence
  drain          - wait until queues are drained


Available params for "set":

  Available params for set checkpoint_param:
    chk_max_items                - Max number of items allowed in a checkpoint.
    chk_period                   - Time bound (in sec.) on a checkpoint.
    item_num_based_new_chk       - true if a new checkpoint can be created based
                                   on.
                                   the number of items in the open checkpoint.
    keep_closed_chks             - true if we want to keep closed checkpoints in
                                   memory.
                                   as long as the current memory usage is below
                                   high water mark.
    max_checkpoints              - Max number of checkpoints allowed per vbucket.
    enable_chk_merge             = True if merging closed checkpoints is enabled.


  Available params for set flush_param:
    access_scanner_enabled       - Enable or disable access scanner task (true/false)
    alog_sleep_time              - Access scanner interval (minute)
    alog_task_time               - Hour in UTC time when access scanner task is
                                   next scheduled to run (0-23).
    backfill_mem_threshold       - Memory threshold (%) on the current bucket quota
                                   before backfill task is made to back off.
    bg_fetch_delay               - Delay before executing a bg fetch (test
                                   feature).
    bfilter_enabled              - Enable or disable bloom filters (true/false)
    bfilter_residency_threshold  - Resident ratio threshold below which all items
                                   will be considered in the bloom filters in full
                                   eviction policy (0.0 - 1.0)
    compaction_exp_mem_threshold - Memory threshold (%) on the current bucket quota
                                   after which compaction will not queue expired
                                   items for deletion.
    compaction_write_queue_cap   - Disk write queue threshold after which compaction
                                   tasks will be made to snooze, if there are already
                                   pending compaction tasks.
    dcp_min_compression_ratio    - Minimum compression ratio of compressed doc against
                                   the original doc. If compressed doc is greater than
                                   this percentage of the original doc, then the doc
                                   will be shipped as is by the DCP producer if value
                                   compression is enabled by the DCP consumer. Applies
                                   to all producers (Ideal range: 0.0 - 1.0)
    defragmenter_enabled         - Enable or disable the defragmenter
                                   (true/false).
    defragmenter_interval        - How often defragmenter task should be run
                                   (in seconds).
    defragmenter_age_threshold   - How old (measured in number of defragmenter
                                   passes) must a document be to be considered
                                   for defragmentation.
    defragmenter_chunk_duration  - Maximum time (in ms) defragmentation task
                                   will run for before being paused (and
                                   resumed at the next defragmenter_interval).
    exp_pager_enabled            - Enable expiry pager.
    exp_pager_stime              - Expiry Pager Sleeptime.
    exp_pager_initial_run_time   - Expiry Pager first task time (UTC)
                                   (Range: 0 - 23, Specify 'disable' to not delay the
                                   the expiry pager, in which case first run will be
                                   after exp_pager_stime seconds.)
    flushall_enabled             - Enable flush operation.
    pager_active_vb_pcnt         - Percentage of active vbuckets items among
                                   all ejected items by item pager.
    max_size                     - Max memory used by the server.
    mem_high_wat                 - High water mark (suffix with '%' to make it a
                                   percentage of the RAM quota)
    mem_low_wat                  - Low water mark. (suffix with '%' to make it a
                                   percentage of the RAM quota)
    mutation_mem_threshold       - Memory threshold (%) on the current bucket quota
                                   for accepting a new mutation.
    timing_log                   - path to log detailed timing stats.
    warmup_min_memory_threshold  - Memory threshold (%) during warmup to enable
                                   traffic
    warmup_min_items_threshold   - Item number threshold (%) during warmup to enable
                                   traffic
    max_num_readers              - Override default number of global threads that
                                   prioritize read operations.
    max_num_writers              - Override default number of global threads that
                                   prioritize write operations.
    max_num_auxio                - Override default number of global threads that
                                   prioritize auxio operations.
    max_num_nonio                - Override default number of global threads that
                                   prioritize nonio operations.

  Available params for "set tap_param":
    tap_keepalive                    - Seconds to hold a named tap connection.
    replication_throttle_queue_cap   - Max disk write queue size to throttle
                                       replication streams ('infinite' means no
                                       cap).
    replication_throttle_cap_pcnt    - Percentage of total items in write queue
```

```
                                             at which we throttle replication input
    replication_throttle_threshold         - Percentage of memory in use to throttle
                                             replication streams.
```

We will specifically be changing the alog_sleep_time for the flush_param. Run the following command from the 3rd node to change the interval for how frequently the access scanner runs to every 1 minute:

```
[ec2-user@ip-172-31-0-128 ~]$ /opt/couchbase/bin/cbepctl
localhost:11210 -b beer-sample set flush_param alog_sleep_time 1
setting param: alog_sleep_time 1
set alog_sleep_time to 1
```

Now, wait about 60 seconds and check to see if the access.log file has been created yet:
First switch to root user:

```
[ec2-user@ip-172-31-0-128 ~]$ sudo –s
```

Search for the access.log on this server:

```
[root@ip-172-31-0-128 ec2-user]# find /opt/couchbase| grep access.log
/opt/couchbase/var/lib/couchbase/data/beer-sample/access.log
```

Note that access.log is a binary file and cannot be opened via a standard linux text editor.
Exit root:

```
[root@ip-172-31-0-128 default]# exit
exit
```

For the 3rd node change the interval for how frequently the access scanner runs to every 60 minutes:

```
[ec2-user@ip-172-31-0-128 ~]$ /opt/couchbase/bin/cbepctl
localhost:11210 -b beer-sample set flush_param alog_sleep_time 60
setting param: alog_sleep_time 60
set alog_sleep_time to 60
```

Next, change the parameter alog_task_time for the 3rd node's beer-sample bucket so that the initial time that the access scanner runs is at 1:00pm UTC:

```
[ec2-user@ip-172-31-0-128 ~]$ /opt/couchbase/bin/cbepctl
localhost:11210 -b beer-sample set flush_param alog_task_time 13
setting param: alog_task_time 13
set alog_task_time to 13
```

These settings now mean that after you install a new Couchbase Server instance or restart the server, by default the scanner will run every 1 hour with a default initial start time of 1:00pm UTC.
This concludes the server warmup section. For more information about handling server warmup settings, visit:

http://docs.couchbase.com/admin/admin/Tasks/tasks-manage-serverWarmup.html

Understanding Views and Indexes:

There are two ways to retrieve information from a Couchbase database: By Key and By View. If you know the key used to store a particular value, then you can use the memcached protocol (or an appropriate memcached compatible client-library) to retrieve the value stored against a specific key. Retrieving data by key is typically the fastest way to pull it out of the Couchbase engine. If you do not know the key, you can use the View system to write a view that outputs the information you need.

A view creates an index on the stored information (typically JSON files) according to the Map and Reduce code defining the view. The view consists of specific fields and information extracted from the objects stored in Couchbase. Views create indexes on your information allowing you to search and select information stored within Couchbase Server.

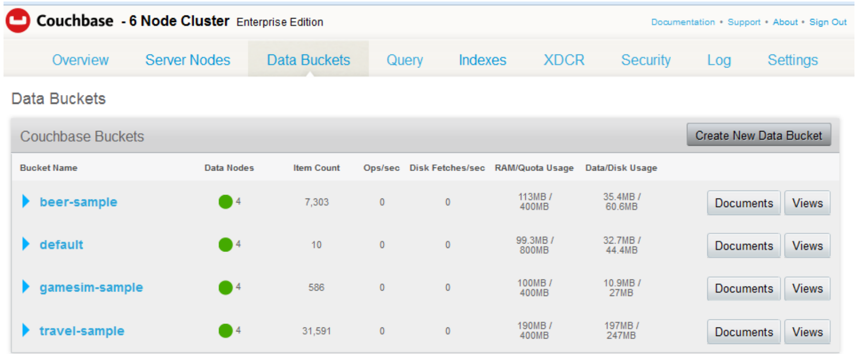
Warning: Views are eventually consistent compared to the underlying stored documents. Documents with expiry times are removed from indexes only when the expiration pager operates to remove the document from the database.
Views can be used within Couchbase Server for a number of reasons, including:

- Indexing and querying data from your stored objects
- Producing lists of data on specific object types
- Producing tables and lists of information based on your stored data
- Extracting or filtering information from the database
- Calculating, summarizing or reducing the information on a collection of stored data

You can create multiple views and therefore multiple indexes and routes into the information stored in your database. By exposing specific fields from the stored information, views enable you to create and query the information stored within your Couchbase Server, perform queries and selection on the information, and paginate through the view output. The View Builder provides an interface for creating your views within the Couchbase Server Web Console.

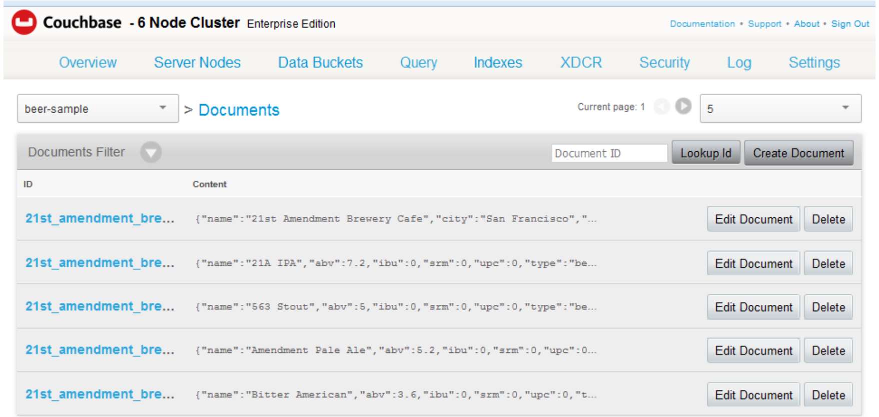
The purpose of a view is take the un-structured, or semi-structured, data stored within your Couchbase Server database, extract the fields and information that you want, and to produce an index of the selected information. Storing information in Couchbase Server using JSON makes the process of selecting individual fields for output easier. The resulting generated structure is a view on the stored data. The view that is created during this process allows you to iterate, select and query the information in your database from the raw data objects that have been stored.
Another important point to keep in mind is that view reads will ALWAYS have a disk seek penalty (unless the OS caches it) as there is no memcached version of the view files.
In this section, we will use the beer-sample bucket to create indexes and views.

Click on Data Buckets in the Web UI and you should currently see 7,303 items in the bucket. Click on 'Documents' next to beer-sample:



Bucket Name	Data Nodes	Item Count	Ops/sec	Disk Fetches/sec	RAM/Quota Usage	Data/Disk Usage	
beer-sample	● 4	7,303	0	0	113MB / 400MB	35.4MB / 60.6MB	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

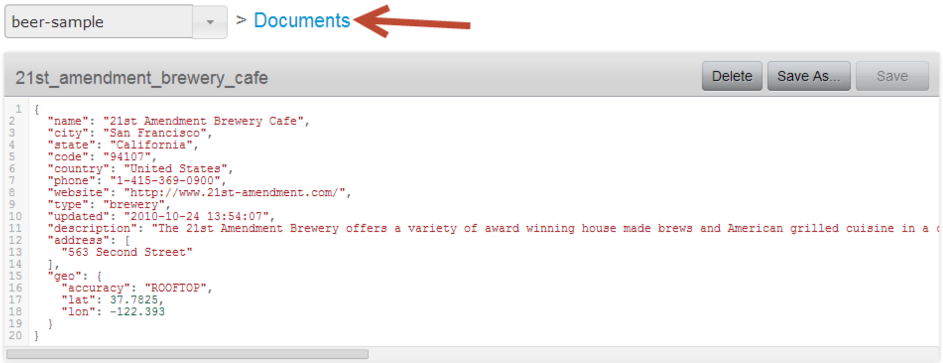
The first 5 items in the beer-sample bucket are now displayed. Click on 'Edit Document' for the first item:



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

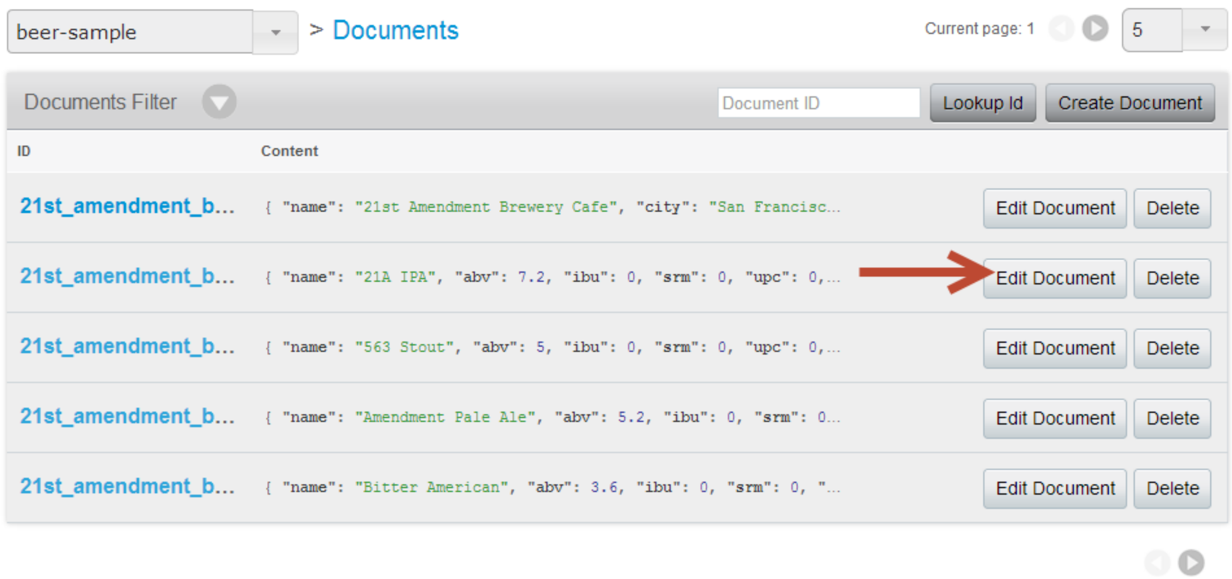
This first JSON document describes the 21st_amendment_brewery_cafe. You can see that the café is located in San Francisco along with its address, geo coordinates and website address.

Click on Documents to return back to the previous screen:



```
{
  "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 c
  "address": [
    "563 Second Street"
  ],
  "geo": {
    "accuracy": "ROOFTOP",
    "lat": 37.7825,
    "lon": -122.393
  }
}
```

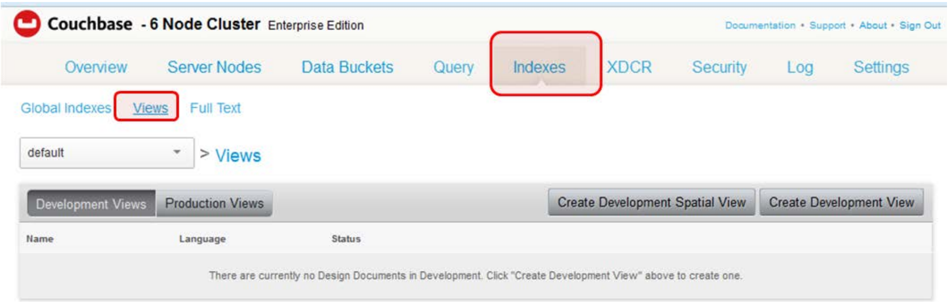
Click on ‘Edit Document’ for the 2nd item:



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

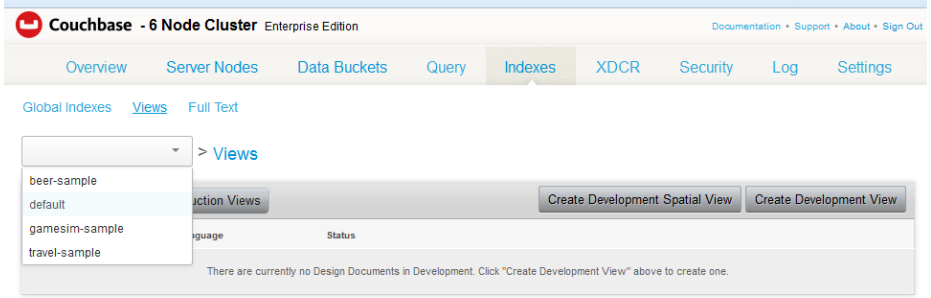
Here you can see details of a specific beer named ‘21A IPA’. The JSON document describes it’s alcohol by value as well under the abv field. Note that most of the items in this bucket are like this one, describing beers rather than breweries.

Click on Indexes at the top then click on Views:



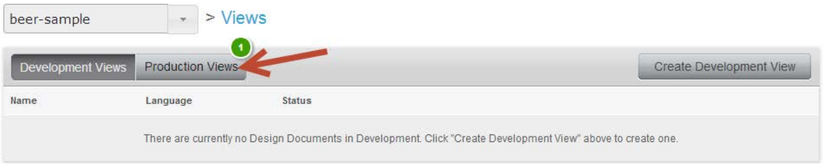
There are currently no Design Documents in Development. Click "Create Development View" above to create one.

Click on the drop-down menu at the top and select the beer-sample bucket:



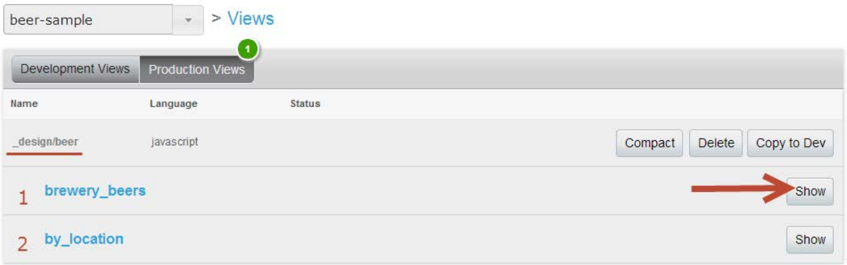
There are currently no Design Documents in Development. Click "Create Development View" above to create one.

By default, there will be 0 development views and 2 production views (under 1 design document) for the beer-sample bucket. Click on 'Production Views' to see the 2 views:



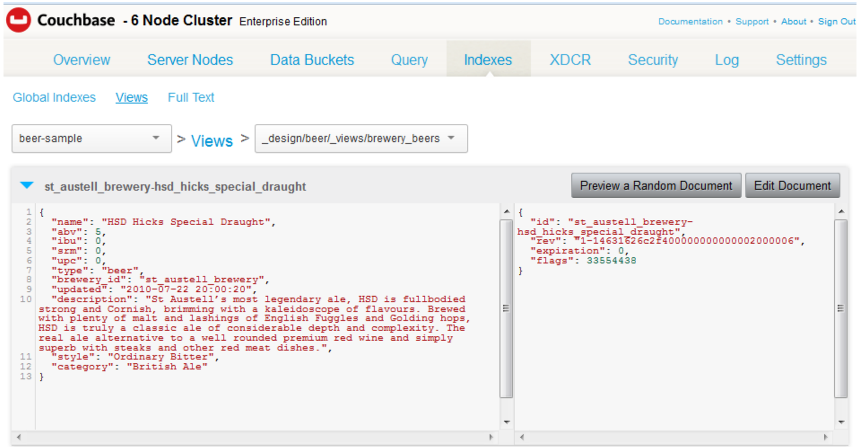
There are currently no Design Documents in Development. Click "Create Development View" above to create one.

There is just 1 design document here named "_design/beer" with 2 views in it: "brewery_beers" and "by_location". Click on Show next to "brewery_beers":



Under this individual view, you will see a random JSON document at the top of the page with the view code and filter results at the bottom.

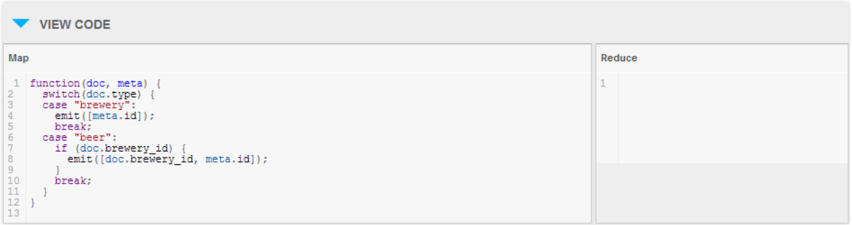
First click on "Preview a Random Document" to display another document:


Couchbase - 6 Node Cluster Enterprise Edition
Documentation • Support • About • Sign Out
Overview
Server Nodes
Data Buckets
Query
Indexes
XDCR
Security
Log
Settings
Global Indexes
Views
Full Text
beer-sample
> Views >
_design/beer/_views/brewery_beers
st_austell_brewery-hsd_hicks_special_draught
Preview a Random Document
Edit Document

```
{
  "name": "HSD Hicks Special Draught",
  "abv": 5,
  "ibu": 0,
  "srm": 0,
  "upc": 0,
  "type": "beer",
  "brewery_id": "st_austell_brewery",
  "updated": "2010-07-22 20:00:20",
  "description": "St Austell's most legendary ale, HSD is fullbodied
strong and Cornish, brimming with a kaleidoscope of flavours. Brewed
with plenty of malt and lashings of English Fuggles and Golding hops,
HSD is truly a classic ale of considerable depth and complexity. The
real ale alternative to a well rounded premium red wine and simply
superb with steaks and other red meat dishes.",
  "style": "Ordinary Bitter",
  "category": "British Ale"
}
```

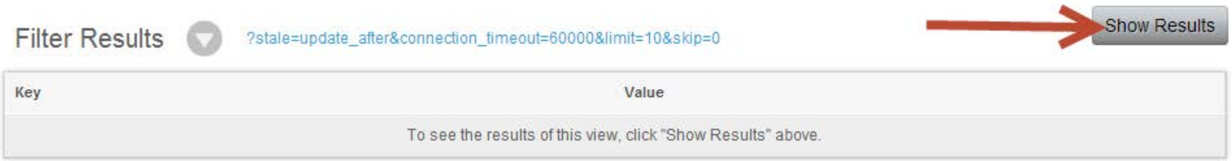
```
{
  "id": "st_austell_brewery-
hsd_hicks_special_draught",
  "rev": "1-14631626c2f400000000000002000006",
  "expiration": 0,
  "flags": 33554438
}
```

The new document that is displayed could be describing a beer or a brewery. Scroll down until you see the ‘View Code’ section. Notice that this specific view has a Map function, but no Reduce function.



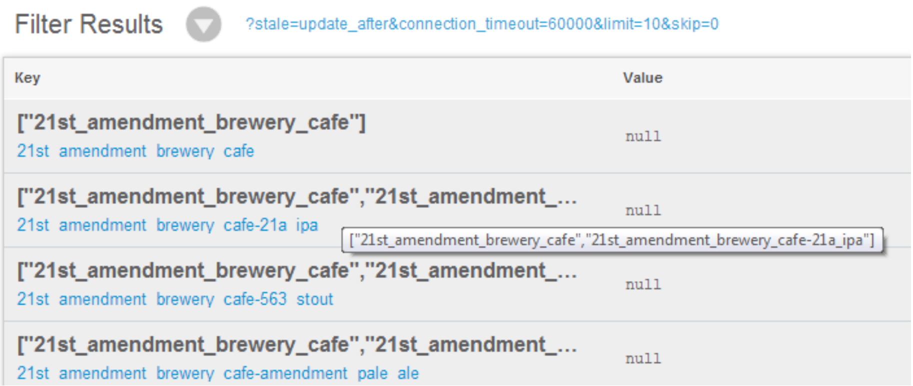
```
function(doc, meta) {
  switch(doc.type) {
  case "brewery":
    emit([meta.id]);
    break;
  case "beer":
    if (doc.brewery_id) {
      emit([doc.brewery_id, meta.id]);
    }
    break;
  }
}
```

Continue scrolling down till you see the “Filter Results” section. Click on “Show Results”:


Filter Results
?stale=update_after&connection_timeout=60000&limit=10&skip=0
Show Results
Key
Value

To see the results of this view, click "Show Results" above.

The results will now be displayed and you should see some results. All results will have null for the value. The first result’s key will name the brewery “21st_amendment_brewery_cafe”. However the second result’s key will be a compound key with 2 components: the brewery that made the beer and the specific beer.


Filter Results
?stale=update_after&connection_timeout=60000&limit=10&skip=0

Key	Value
["21st_amendment_brewery_cafe"] 21st_amendment_brewery_cafe	null
["21st_amendment_brewery_cafe","21st_amendment_... 21st_amendment_brewery_cafe-21a_ipa	null
["21st_amendment_brewery_cafe","21st_amendment_... 21st_amendment_brewery_cafe-563_stout	null
["21st_amendment_brewery_cafe","21st_amendment_... 21st_amendment_brewery_cafe-amendment_pale_ale	null

["21st_amendment_brewery_cafe","21st_amendment_brewery_cafe-21a_ipa"]

Scroll up a bit to take a closer look at the Map code:

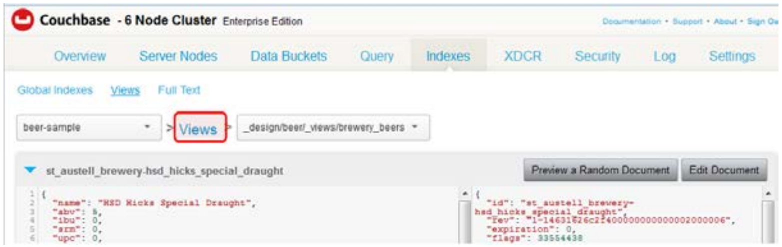
```
function(doc, meta) {
  switch(doc.type) {
  case "brewery":
    emit([meta.id]);
    break;
  case "beer":
    if (doc.brewery_id) {
      emit([doc.brewery_id, meta.id]);
    }
    break;
  }
}
```

This code will simply scan through all 7,303 items in the beer-sample bucket and emit just the document id (key) if the document is describing a brewery or emit the brewery id and the document id (key) if the document is describing a beer. Under production it is not possible to run the View against just a smaller subset of the items.
It is also not possible to edit the code in this view since it is a production view. You have to copy it to a development view before you can edit it. Let’s do that now.

Create a custom View under Dev and publish it to Production:

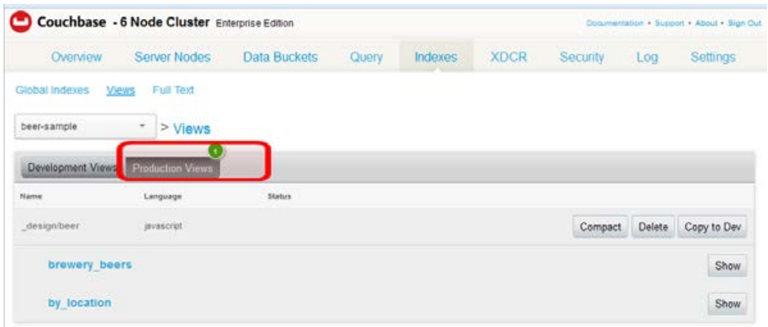
Let’s assume that as a Couchbase Admin you have been handed some Map and Reduce code by a developer who would like you to run it against the beer-sample bucket. We will now walk through the process of creating a custom View and publishing it to production.

Click on Views and then production Views:


Couchbase - 6 Node Cluster Enterprise Edition
Documentation • Support • About • Sign Out
Overview
Server Nodes
Data Buckets
Query
Indexes
XDCR
Security
Log
Settings
Global Indexes
Views
Full Text
beer-sample
> Views >
_design/beer/_views/brewery_beers
st_austell_brewery-hsd_hicks_special_draught
Preview a Random Document
Edit Document

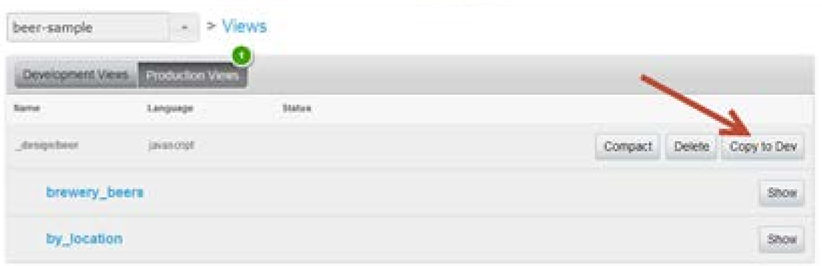
```
{
  "name": "HSD Hicks Special Draught",
  "abv": 5,
  "ibu": 0,
  "srm": 0,
  "upc": 0,
```

```
{
  "id": "st_austell_brewery-
hsd_hicks_special_draught",
  "rev": "1-14631826c2f4000000000000002000006",
  "expiration": 0,
  "flags": 33554438
```



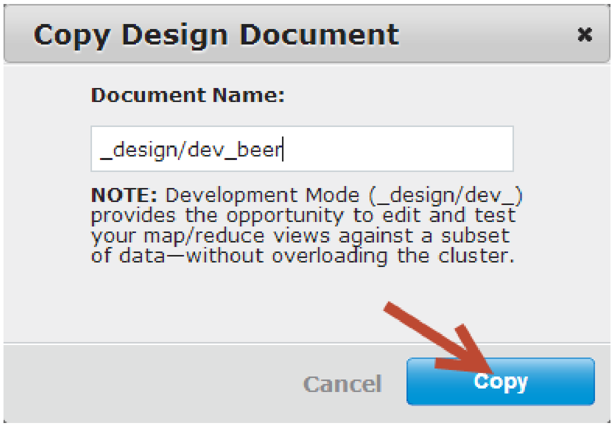
Name	Language	Status
_design/beer	javascript	
brewery_beers		
by_location		

Click 'Copy to Dev' to copy the entire sample Design Document with both views into Development:



Name	Language	Status
_design/beer	javascript	
brewery_beers		
by_location		

Name for the Design Doc. Go with the default suggestion and click Copy:

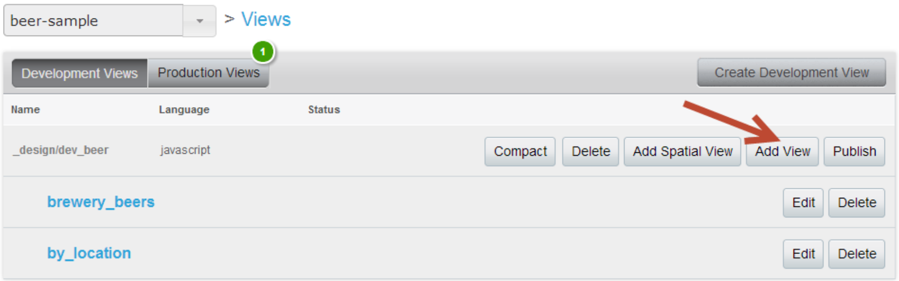


Copy Design Document

Document Name: _design/dev_beer

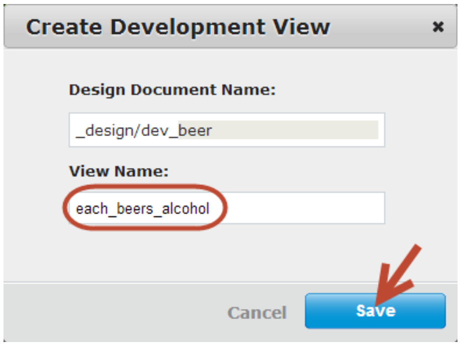
NOTE: Development Mode (_design/dev_) provides the opportunity to edit and test your map/reduce views against a subset of data—without overloading the cluster.

The Design Doc now appears under Development Views. Let's create a new View under this design document. Click on "Add View":



Name	Language	Status
_design/dev_beer	javascript	
brewery_beers		
by_location		

On the popup, name the view "each_beers_alcohol" and click on Save:

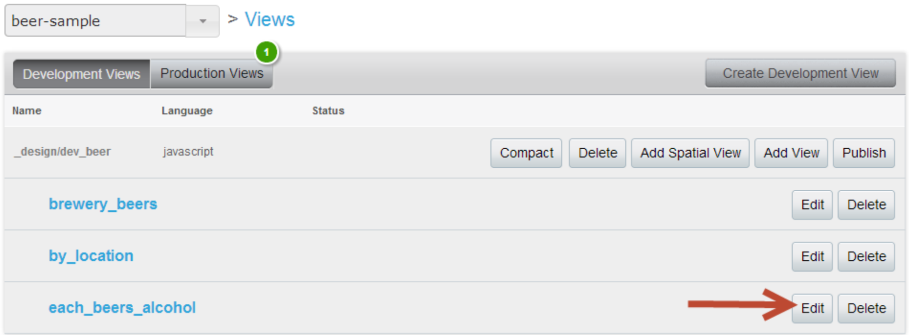


Create Development View

Design Document Name: _design/dev_beer

View Name: each_beers_alcohol

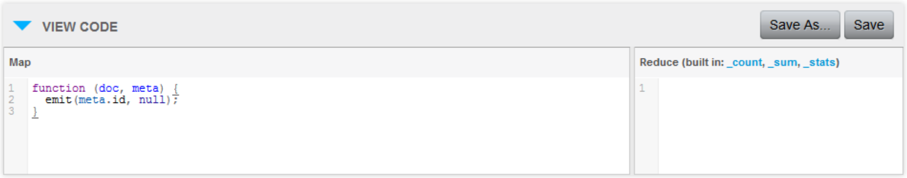
The new View appears in the GUI. Click on Edit next to the new View:



beer-sample > Views

Name	Language	Status	
_design/dev_beer	javascript		Compact Delete Add Spatial View Add View Publish
brewery_beers			Edit Delete
by_location			Edit Delete
each_beers_alcohol			Edit Delete

Scroll down till you see the code and notice that the view comes with some default code. This Map code is going to simply emit each document's ID as the key and null for the value:

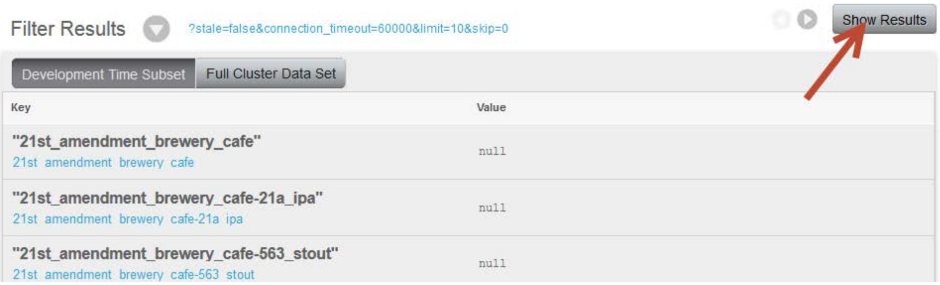


Map

```
function (doc, meta) {
  emit(meta.id, null);
}
```

Reduce (built in: _count, _sum, _stats)

Verify that the View will emit what you suspect it'll emit by clicking on Show Results:



Filter Results ?stale=false&connection_timeout=60000&limit=10&skip=0

Key	Value
"21st_amendment_brewery_cafe" 21st amendment brewery cafe	null
"21st_amendment_brewery_cafe-21a_ipa" 21st amendment brewery cafe-21a ipa	null
"21st_amendment_brewery_cafe-563_stout" 21st amendment brewery cafe-563 stout	null

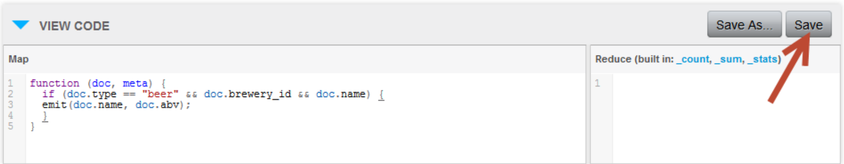
Indeed all the values are null and the key is just each original document's keys.

Scroll up a bit to the "View Code" section and copy the following custom code into Map(replacing the first 3 lines of code):

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

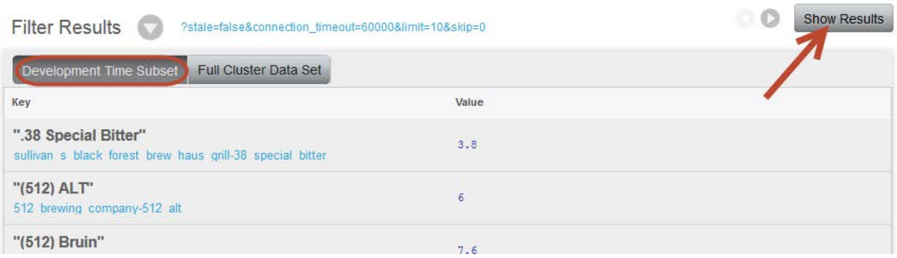
The above map code will create an index of beer names (doc.name) and the alcohol by volume values (doc.abv). It will first filter the JSON documents for doc.type == beer where the doc.brewery_id is non-null and the doc.name is non-null. Then it will output as a key the beer name (doc.name) which is searchable and the beer alcohol by value (doc.abv) which is the value.
The Reduce code section will remain empty.

Click Save:



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

Notice under 'Filter Results' that the default setting will only run the Indexer for the View against a subset of the JSON docs. Leave the default alone and click on Show Results to test the View:



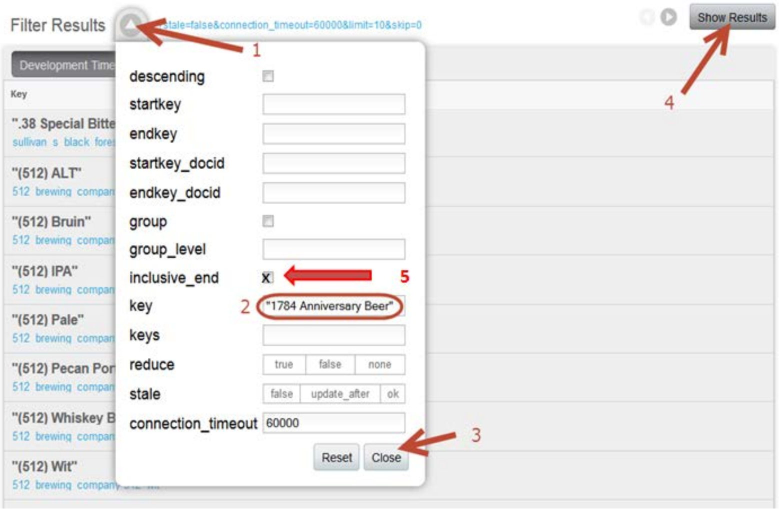
?stale=false&connection_timeout=60000&limit=10&skip=0

Key	Value
".38 Special Bitter" sullivan s black forest brew haus grill-38 special bitter	3.8
"(512) ALT" 512 brewing company-512 alt	6
"(512) Bruin"	7.6

In the results, you should see a specific beer type under the key and the alcohol content under Value.
It is also possible to filter the results to look for specific items. For example, what if you wanted to see the alcohol content for a beer named "1784 Anniversary Beer"?

Warning: In the next step, do not copy + paste "1784 Anniversary Beer" as the double quotes may not copy as standard ascii!

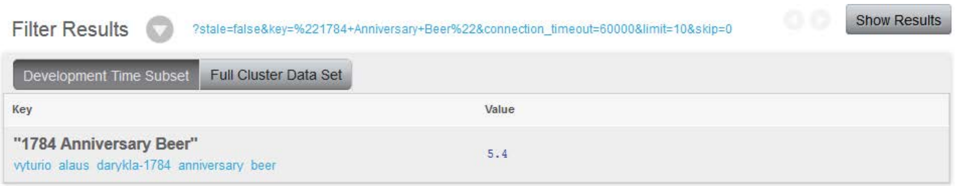
Click on the arrow next to "Filter Results" to see a dropdown of filtering options. Type in "1784 Anniversary Beer" for the key and check "inclusive_end" box , click Close. Then click on "Show Results":



?stale=false&connection_timeout=60000&limit=10&skip=0

descending	□
startkey	
endkey	
startkey_docid	
endkey_docid	
group	□
group_level	
inclusive_end	☒
key	"1784 Anniversary Beer"
keys	
reduce	true / false / none
stale	false / update_after / ok
connection_timeout	60000

The Result for just the specific beer that was requested is displayed:

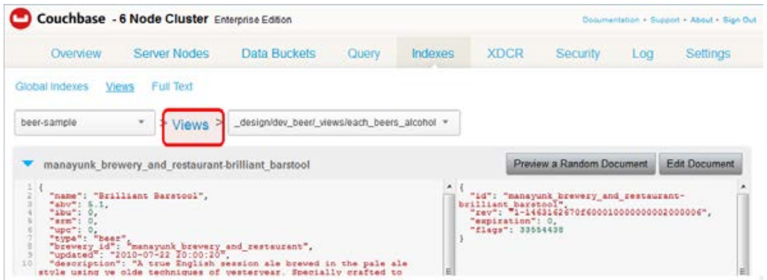


?stale=false&key=%221784+Anniversary+Beer%22&connection_timeout=60000&limit=10&skip=0

Key	Value
"1784 Anniversary Beer" vyturio alaus darykla-1784 anniversary beer	5.4

We are finished creating our first custom view. Scroll back to the top of the page and

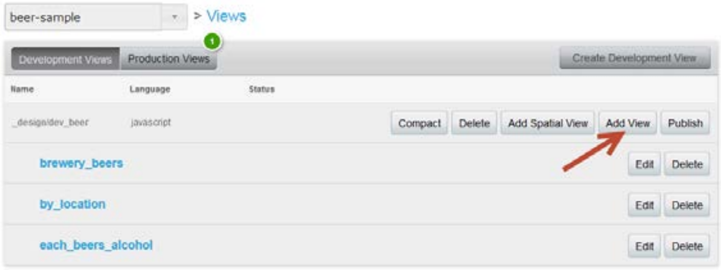
click on Views:


Couchbase - 6 Node Cluster Enterprise Edition
Documentation • Support • About • Sign Out
Overview
Server Nodes
Data Buckets
Query
Indexes
XDCR
Security
Log
Settings
Global Indexes
Views
Full Text
beer-sample
Views
_design/dev_beer/_views/each_beers_alcohol
manayunk_brewery_and_restaurant-brilliant_barstool
Preview a Random Document
Edit Document

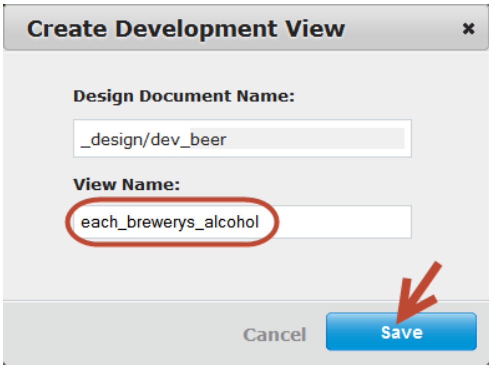
```
{
  "name": "Brilliant Barstool",
  "abv": 5.1,
  "ibu": 0,
  "srm": 0,
  "upc": 0,
  "type": "beer",
  "brewery_id": "manayunk_brewery_and_restaurant",
  "updated": "2010-07-22 20:00:20",
  "description": "A true English session ale brewed in the pale ale
style using ye olde techniques of yesteryear. Specially crafted to
```

```
{
  "id": "manayunk_brewery_and_restaurant-
brilliant_barstool",
  "rev": "1-1463162670f60001000000002000006",
  "expiration": 0,
  "flags": 33554438
}
```

Click on "Add View" to create another custom View:



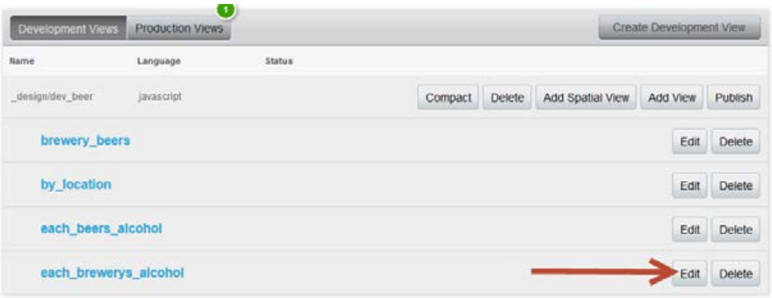
In the pop-up, name the view "each_brewerys_alcohol" and click Save:


Create Development View

Design Document Name: _design/dev_beer

View Name: each_brewerys_alcohol

Click on Edit next to the newly created View:



In this view, we want to see each statistics about each brewery such as how many beers it has and out of all of the beers a brewery may have, what is the minimum alcohol content and what is the maximum alcohol content.

In order to achieve this, copy + paste the following Map function and Reduce function into the new View and then click Save:

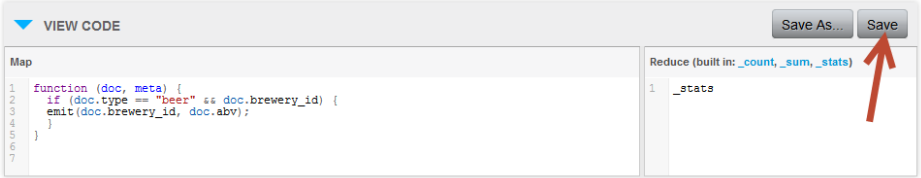
Map:

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

Reduce:

```
_stats
```

Then click save



Map

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

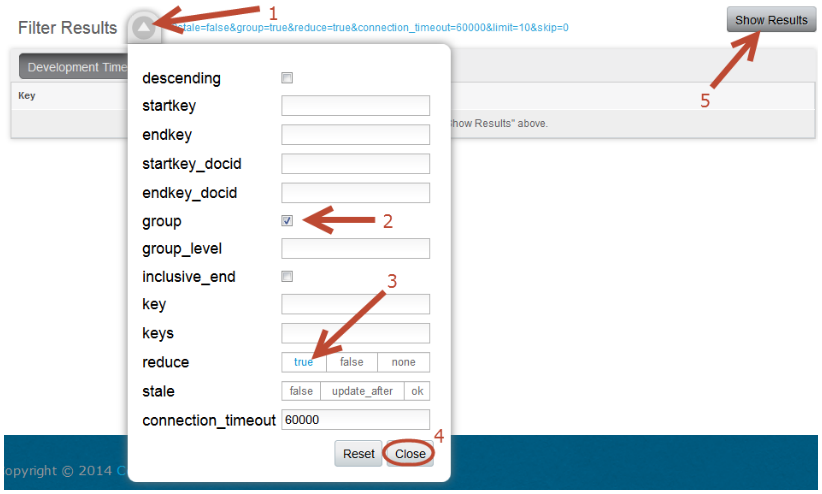
Reduce (built in: _count, _sum, _stats)

```
_stats
```

Under Filter Results, click on the arrow to open the drop down and:

- **place a check next to group**
- **click true for reduce**
- **click close**

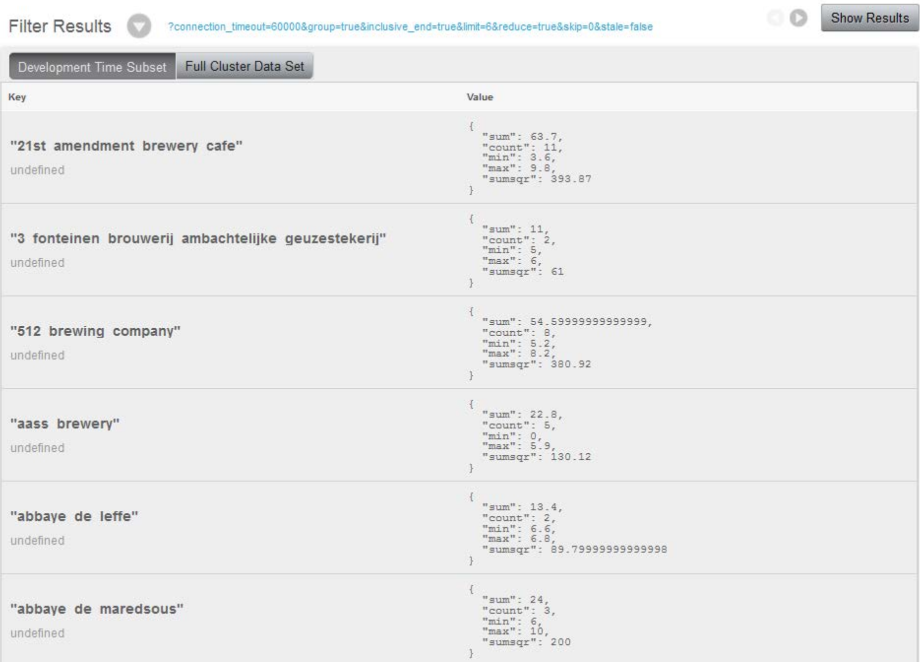
Then click "Show Results":



?stale=false&group=true&reduce=true&connection_timeout=60000&limit=10&skip=0

descending □
startkey
endkey
startkey_docid
endkey_docid
group ☑
group_level
inclusive_end □
key
keys
reduce [✓ true] false none
stale false update_after ok
connection_timeout 60000

The results will show you each brewery as the key and in the value you will see the count of how many beers the brewery has along with the min and max values for beer at the brewery:



?connection_timeout=60000&group=true&inclusive_end=true&limit=6&reduce=true&skip=0&stale=false

Key	Value

"21st amendment brewery cafe"
undefined
	{
	"sum": 63.7,
	"count": 11,
	"min": 3.6,
	"max": 9.8,
	"sumsqr": 393.87
	}

"3 fonteinen brouwerij ambachtelijke geuzestekerij"
undefined
	{
	"sum": 11,
	"count": 2,
	"min": 5,
	"max": 6,
	"sumsqr": 61
	}

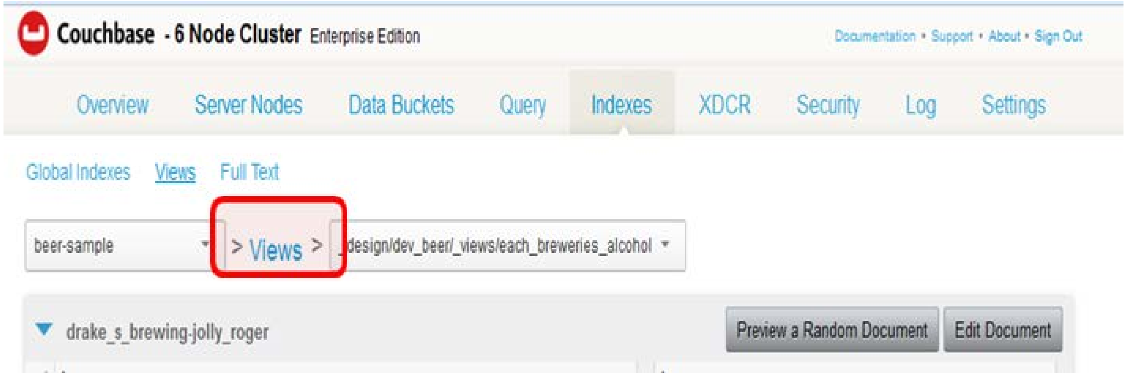
"512 brewing company"
undefined
	{
	"sum": 54.59999999999999,
	"count": 8,
	"min": 5.2,
	"max": 8.2,
	"sumsqr": 380.92
	}

"aass brewery"
undefined
	{
	"sum": 22.8,
	"count": 5,
	"min": 0,
	"max": 5.9,
	"sumsqr": 130.12
	}

"abbaye de leffe"
undefined
	{
	"sum": 13.4,
	"count": 2,
	"min": 6.6,
	"max": 6.8,
	"sumsqr": 89.79999999999998
	}

"abbaye de maredsous"
undefined
	{
	"sum": 24,
	"count": 3,
	"min": 6,
	"max": 10,
	"sumsqr": 200
	}

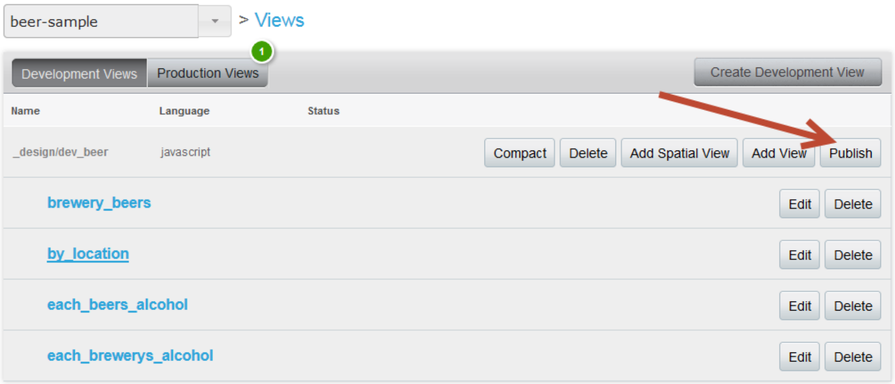
We are now finished creating our 2 custom views. It's now time to publish them to Production. Click on Views at the top:



beer-sample > Views > _design/dev_beer/_views/each_breweries_alcohol

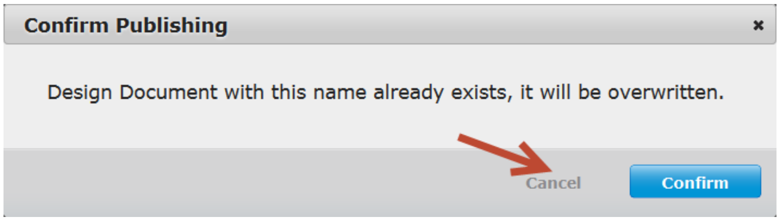
drake_s_brewing-jolly_roger

Click on Publish:



A pop-up appears reminding you that there is already a production design document with the same name and if we publish this, it will overwrite the older production doc.

Click on Cancel:

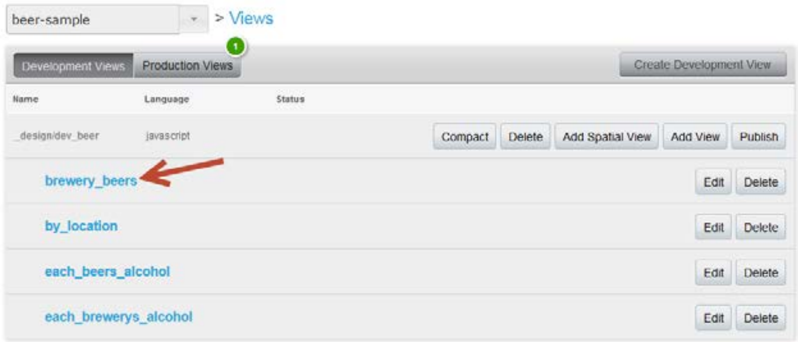


Design Document with this name already exists, it will be overwritten.

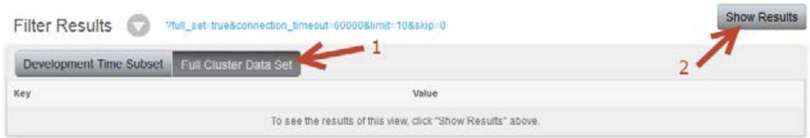
This is a good time to think about what would happen if you had clicked Confirm. It would have replaced the production design doc with the new one. However, after publishing the design doc, it would have still taken some time to generate the four indexes/views against the full production data set for the beer-sample bucket.

What we would like to do instead is pre-generate the 4 indexes/views while in Development mode and then when we publish to Production, the indexes/views should be ready to query against right away. The way to do this is the simply run the 4 views against the 'Full Cluster Data Set'.

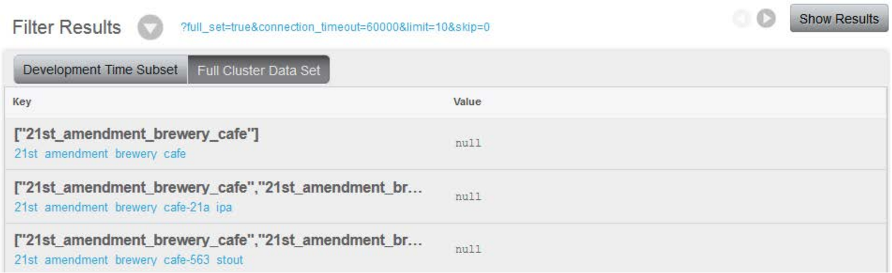
Click on the brewery_beers view:



Click on "Full Cluster Data Set" then "Show Results":

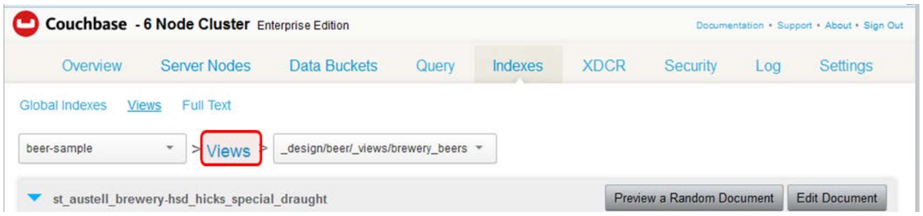

Filter Results
?full_set=true&connection_timeout=60000&limit=10&skip=0
Show Results
Development Time Subset
Full Cluster Data Set
1
2
Key
Value
To see the results of this view, click "Show Results" above.

The index for this specific view is now generated. When we publish this index to production, the md5 hash for this development index will match the md5 hash for the production index and therefore the production job will use the same dev index that we already generated results for.

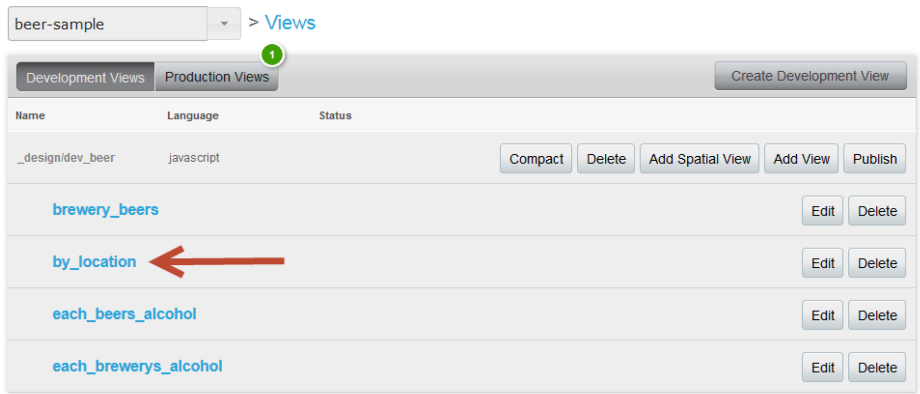


Key	Value
["21st_amendment_brewery_cafe"] 21st_amendment_brewery_cafe	null
["21st_amendment_brewery_cafe","21st_amendment_br... 21st_amendment_brewery_cafe-21a_ipa	null
["21st_amendment_brewery_cafe","21st_amendment_br... 21st_amendment_brewery_cafe-563_stout	null

Click on Views:


Couchbase - 6 Node Cluster Enterprise Edition
Documentation • Support • About • Sign Out
Overview
Server Nodes
Data Buckets
Query
Indexes
XDCR
Security
Log
Settings
Global Indexes
Views
Full Text
beer-sample
> Views
> _design/beer/_views/brewery_beers
st_austell_brewery-hsd_hicks_special_draught
Preview a Random Document
Edit Document

Click on the by_location view:



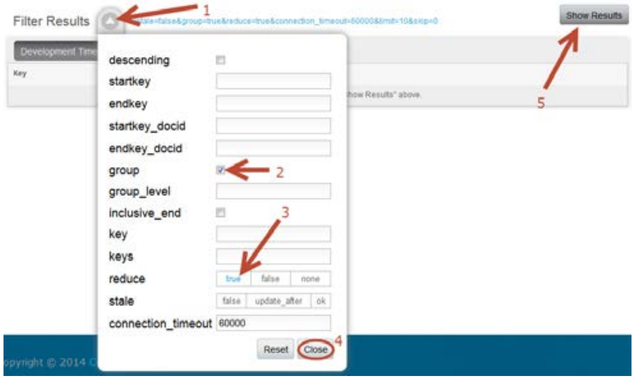
beer-sample > Views

Name	Language	Status	
_design/dev_beer	javascript		Compact Delete Add Spatial View Add View Publish
brewery_beers			Edit Delete
by_location			Edit Delete
each_beers_alcohol			Edit Delete
each_brewerys_alcohol			Edit Delete

Under Filter Results, click on the arrow to open the drop down and:

- **place a check next to group**
- **click true for reduce**
- **click close**

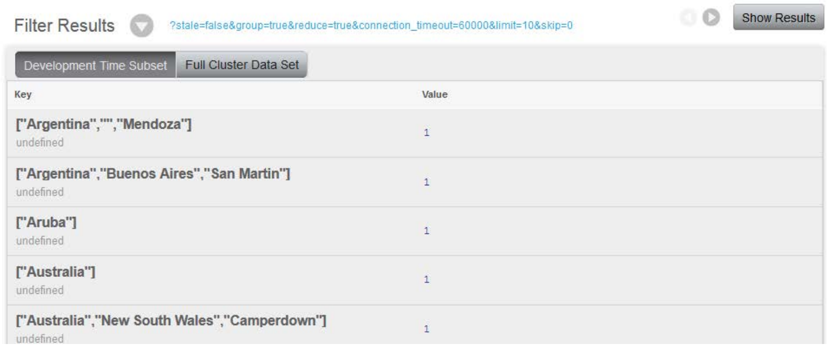
Then click "Show Results":



Filter Results ?stale=false&group=true&reduce=true&connection_timeout=60000&limit=10&skip=0

- descending ☐
- startkey
- endkey
- startkey_docid
- endkey_docid
- group ☑
- group_level
- inclusive_end ☐
- key
- keys
- reduce [✓ true] false none
- stale false update_after ok
- connection_timeout 60000

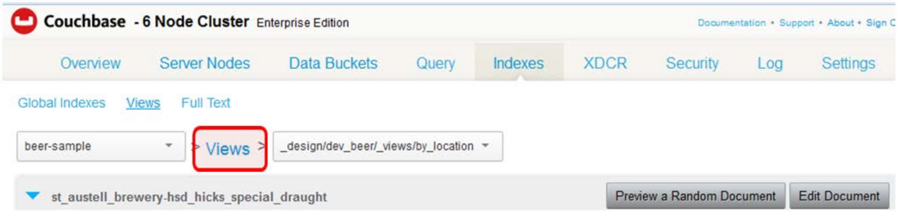
The index for this specific view is also now generated.



Filter Results ?stale=false&group=true&reduce=true&connection_timeout=60000&limit=10&skip=0

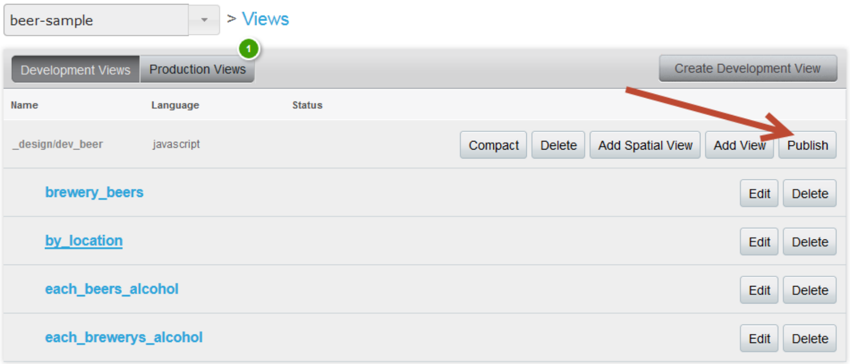
Key	Value
["Argentina","","Mendoza"] undefined	1
["Argentina","Buenos Aires","San Martin"] undefined	1
["Aruba"] undefined	1
["Australia"] undefined	1
["Australia","New South Wales","Camperdown"] undefined	1

Click on Views at the top:


Couchbase - 6 Node Cluster Enterprise Edition
Documentation • Support • About • Sign O
Overview
Server Nodes
Data Buckets
Query
Indexes
XDCR
Security
Log
Settings
Global Indexes
Views
Full Text
beer-sample
> Views >
_design/dev_beer/_views/by_location
st_austell_brewery-hsd_hicks_special_draught
Preview a Random Document
Edit Document

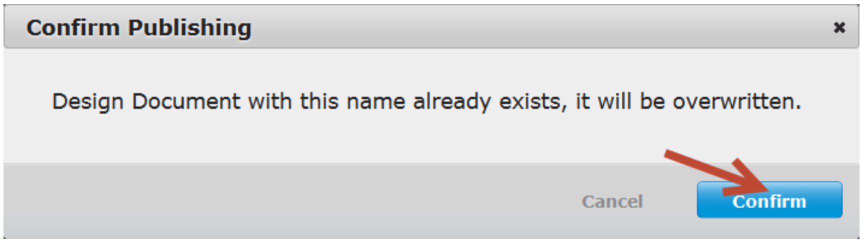
Instead of pre-generating Indexes for the 'each_breers_alcohol' and 'each_brewerys_alcohol' views, let's just go ahead and publish the entire design document to Production. The indexes for these two views will be created lazily later by the production indexer.

Click on Publish again:



Name	Language	Status	
_design/dev_beer	javascript		Compact Delete Add Spatial View Add View Publish
brewery_beers			Edit Delete
by_location			Edit Delete
each_beers_alcohol			Edit Delete
each_brewerys_alcohol			Edit Delete

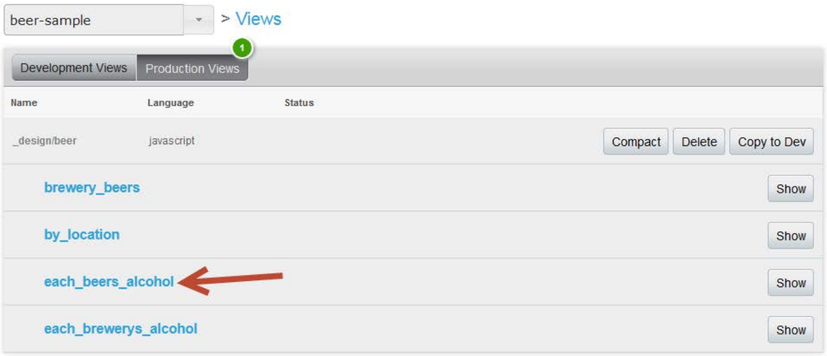
This time, click on Confirm:


Confirm Publishing

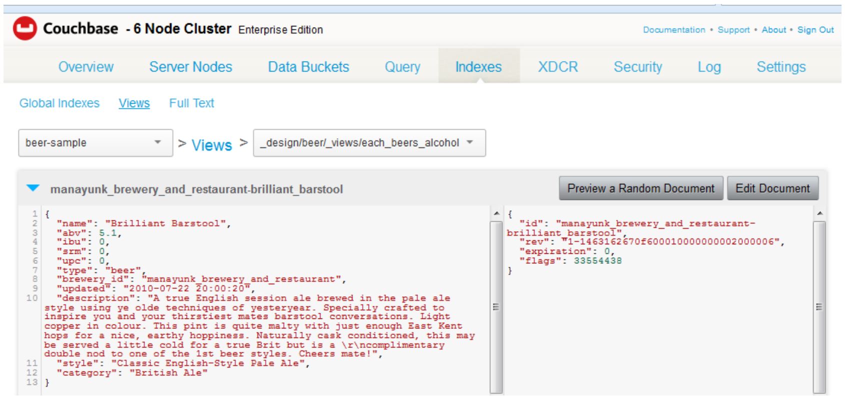
Design Document with this name already exists, it will be overwritten.

You will now see the Design Document with 4 views under Production.

Click on the view for 'each_beers_alcohol':



You should now see the 'each_beers_alcohol' view. Keep this browser tab open and continue to the next section.


Couchbase - 6 Node Cluster Enterprise Edition
Documentation • Support • About • Sign Out
Overview
Server Nodes
Data Buckets
Query
Indexes
XDCR
Security
Log
Settings
Global Indexes
Views
Full Text
beer-sample
> Views >
_design/beer/_views/each_beers_alcohol
manayunk_brewery_and_restaurant-brilliant_barstool
Preview a Random Document
Edit Document

```
{
  "name": "Brilliant Barstool",
  "abv": 5.1,
  "ibu": 0,
  "srm": 0,
  "upc": 0,
  "type": "beer",
  "brewery_id": "manayunk_brewery_and_restaurant",
  "updated": "2010-07-22 20:00:20",
  "description": "A true English session ale brewed in the pale ale
style using ye olde techniques of yesteryear. Specially crafted to
inspire you and your thirstiest mates barstool conversations. Light
copper in colour. This pint is quite malty with just enough East Kent
hops for a nice, earthy hoppiness. Naturally cask conditioned, this may
be served a little cold for a true Brit but is a \r\ncomplimentary
double nod to one of the 1st beer styles. Cheers mate!",
  "style": "Classic English-Style Pale Ale",
  "category": "British Ale"
}
```

```
{
  "id": "manayunk_brewery_and_restaurant-
brilliant_barstool",
  "rev": "1-1463162670f6000100000000002000006",
  "expiration": 0,
  "flags": 33554438
}
```

Viewing analytics and metrics for the beer-sample views and compaction:

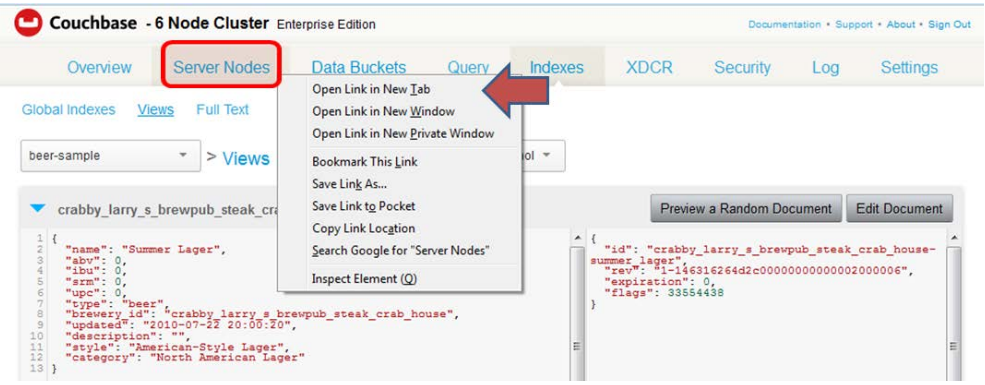
Couchbase's Web UI displays 3 statistics for each design document:

- actual data size
- data size on disk (includes overhead for updates and deletes)
- read operations per second for the design doc

In this section we will generate some fake load against the 'each_beers_alcohol' view and watch the metrics in the Web UI.

You will need two tabs open into the Couchbase web UI to accomplish this. The first tab will have the screen for the specific 'each_beers_alcohol' open. The second tab will display the metrics and statistics for the beer-sample bucket.

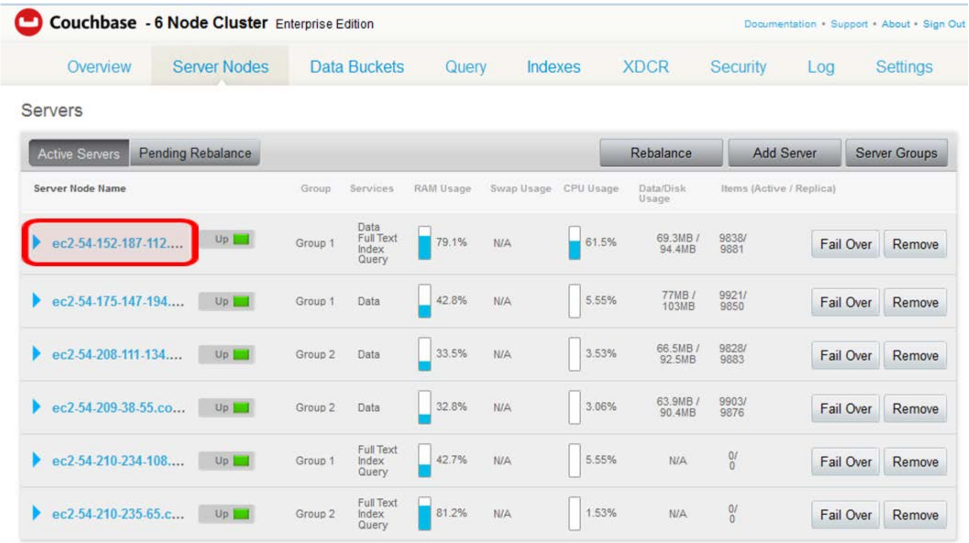
So, keep the existing tab open and right-click on "Server Nodes" at the top and click "Open Link in New Tab":



```
{
  "name": "Summer Lager",
  "abv": 0,
  "ibu": 0,
  "srm": 0,
  "upc": 0,
  "type": "beer",
  "brewery_id": "crabby_larry_s_brewpub_steak_crab_house",
  "updated": "2010-07-22 20:00:20",
  "description": "",
  "style": "American-Style Lager",
  "category": "North American Lager"
}
```

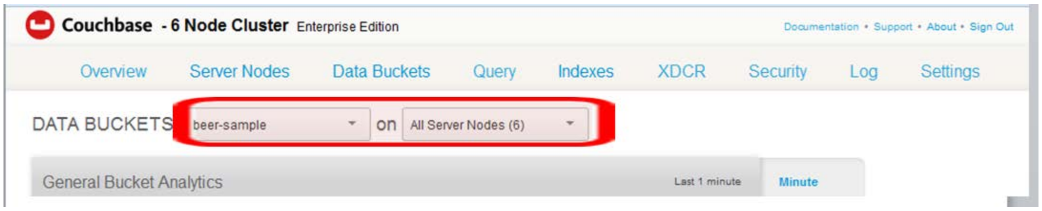
```
{
  "id": "crabby_larry_s_brewpub_steak_crab_house-
summer_lager",
  "rev": "1-146316264d2c0000000000002000006",
  "expiration": 0,
  "flags": 33554438
}
```

In the new tab, click on the first server in the list:

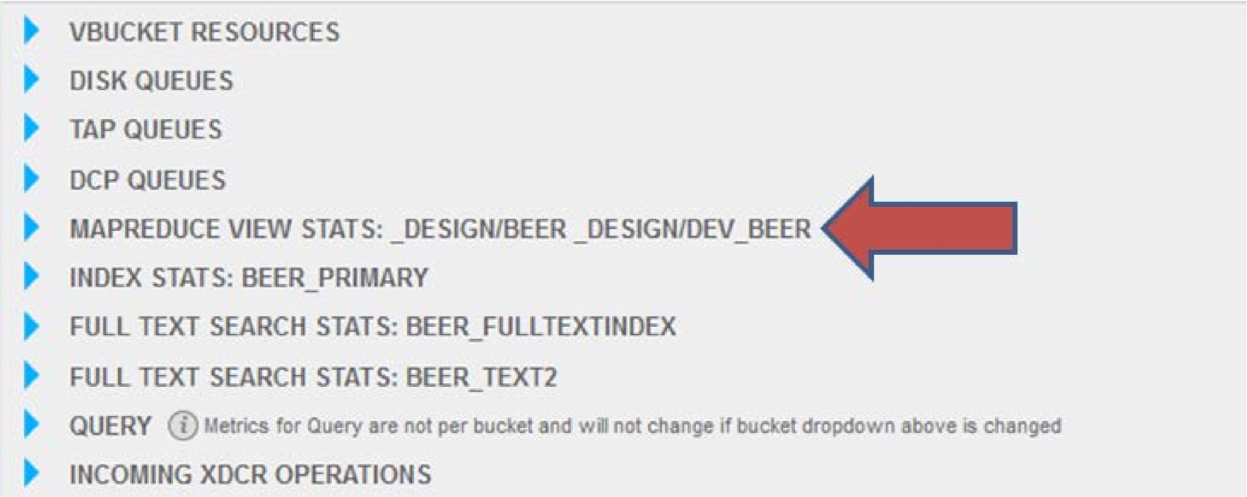


Server Node Name		Group	Services	RAM Usage	Swap Usage	CPU Usage	Data/Disk Usage	Items (Active / Replica)		
ec2-54-152-187-112....	Up	Group 1	Data Full Text Index Query	79.1%	N/A	61.5%	69.3MB / 94.4MB	9838/ 9881	Fail Over	Remove
ec2-54-175-147-194....	Up	Group 1	Data	42.8%	N/A	5.55%	77MB / 103MB	9921/ 9850	Fail Over	Remove
ec2-54-208-111-134....	Up	Group 2	Data	33.5%	N/A	3.53%	66.5MB / 92.5MB	9828/ 9883	Fail Over	Remove
ec2-54-209-38-55.co...	Up	Group 2	Data	32.8%	N/A	3.06%	63.9MB / 90.4MB	9903/ 9876	Fail Over	Remove
ec2-54-210-234-108....	Up	Group 1	Full Text Index Query	42.7%	N/A	5.55%	N/A	0/ 0	Fail Over	Remove
ec2-54-210-235-65.c...	Up	Group 2	Full Text Index Query	81.2%	N/A	1.53%	N/A	0/ 0	Fail Over	Remove

In the drop-down at the top, choose the 'beer-sample' bucket and choose to view metrics on 'All Server Nodes':

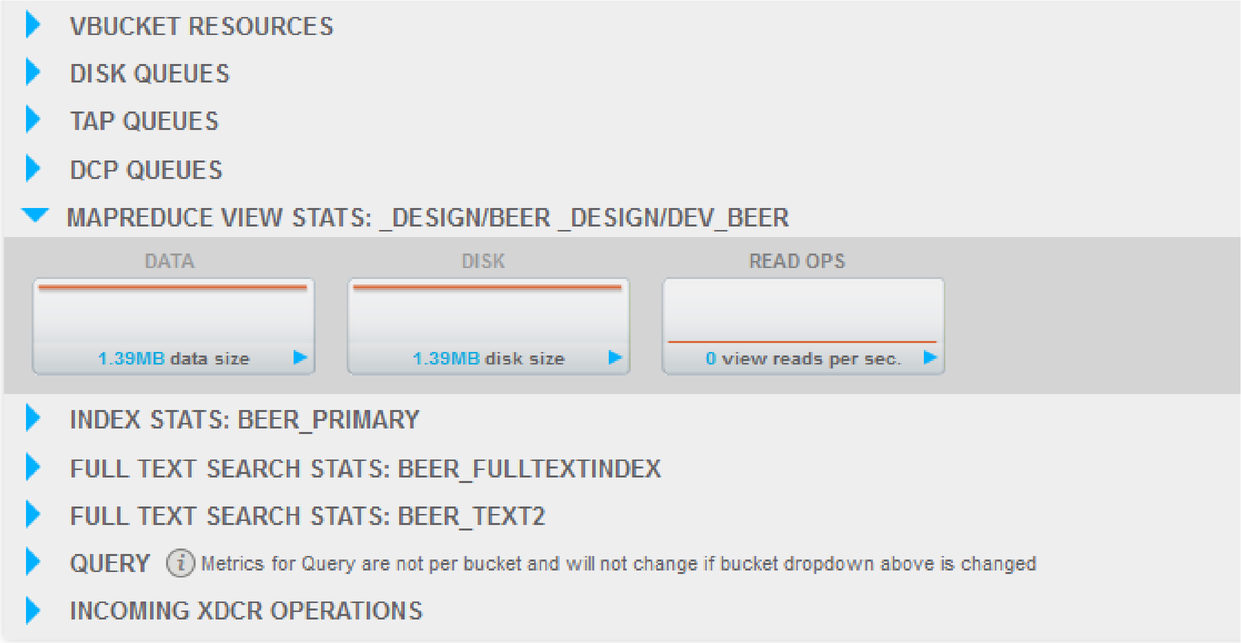


Scroll down and click the arrow next the "MAPREDUCE VIEW STATS: _DESIGN/BEER _DESIGN/DEV_BEER":



VBUCKET RESOURCES
DISK QUEUES
TAP QUEUES
DCP QUEUES
MAPREDUCE VIEW STATS: _DESIGN/BEER _DESIGN/DEV_BEER
INDEX STATS: BEER_PRIMARY
FULL TEXT SEARCH STATS: BEER_FULLTEXTINDEX
FULL TEXT SEARCH STATS: BEER_TEXT2
QUERY ⓘ Metrics for Query are not per bucket and will not change if bucket dropdown above is changed
INCOMING XDCR OPERATIONS

The 3 metrics for the design document are now displayed:



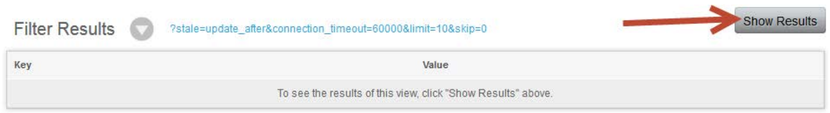
VBUCKET RESOURCES
DISK QUEUES
TAP QUEUES
DCP QUEUES
MAPREDUCE VIEW STATS: _DESIGN/BEER _DESIGN/DEV_BEER

DATA	DISK	READ OPS
1.39MB data size	1.39MB disk size	0 view reads per sec.

INDEX STATS: BEER_PRIMARY
FULL TEXT SEARCH STATS: BEER_FULLTEXTINDEX
FULL TEXT SEARCH STATS: BEER_TEXT2
QUERY ⓘ Metrics for Query are not per bucket and will not change if bucket dropdown above is changed
INCOMING XDCR OPERATIONS

Notice that the "Read Ops" is currently 0. Let's generate some small load against this view to increase the Read Ops counter.

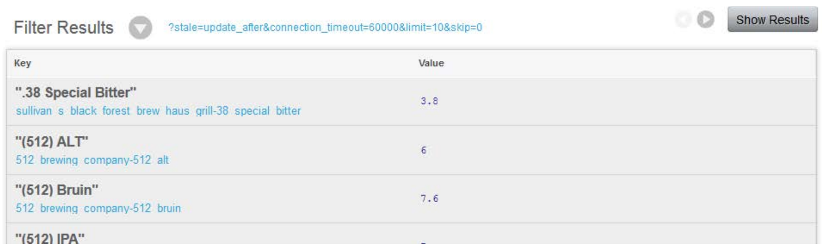
Switch back to the tab displaying the 'each_beers_alcohol' View and at bottom of the page, click on "Show Results":



Filter Results ?stale=update_after&connection_timeout=60000&limit=10&skip=0 Show Results

Key	Value
To see the results of this view, click "Show Results" above.	

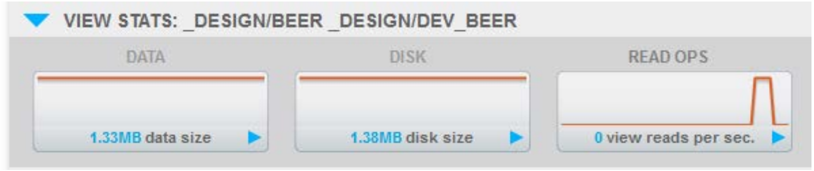
The results will be displayed:



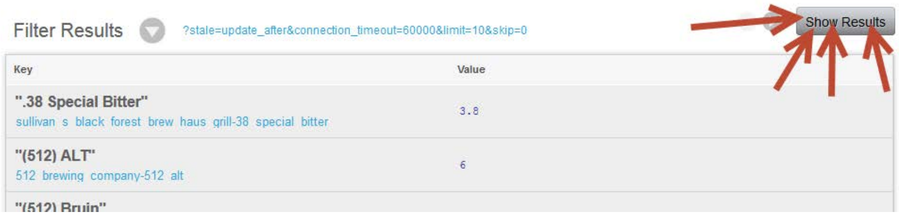
Filter Results ?stale=update_after&connection_timeout=60000&limit=10&skip=0 Show Results

Key	Value
".38 Special Bitter" sullivan_s_black_forest_brew_haus_grill-38_special_bitter	3.8
"(512) ALT" 512_brewing_company-512_alt	6
"(512) Bruin" 512_brewing_company-512_bruin	7.6
"(512) IPA"	

Quickly, switch back to the other tab with the metrics and notice that the Read Ops chart has a small spike:



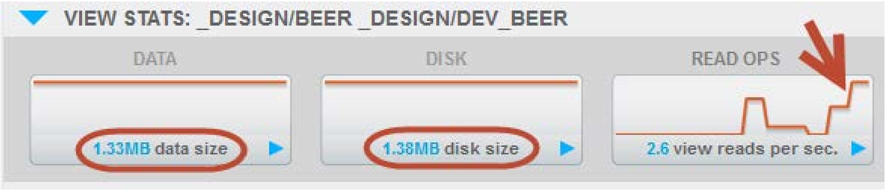
Once again, go to the tab with the 'each_beers_alcohol' view and this time repeatedly and quickly click on "Show Results" 7 – 10 times:



Filter Results ?stale=update_after&connection_timeout=60000&limit=10&skip=0 Show Results

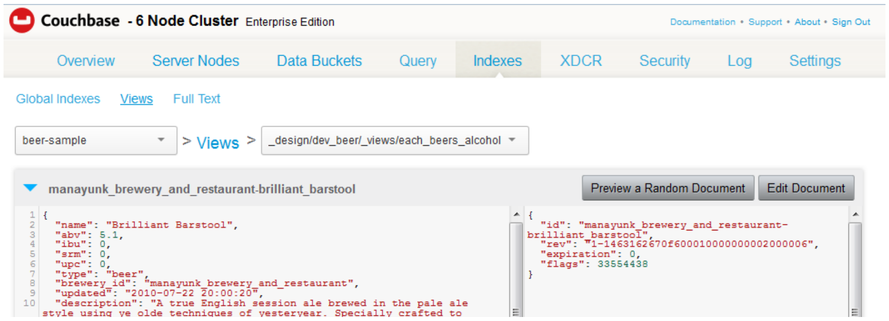
Key	Value
".38 Special Bitter" sullivan_s_black_forest_brew_haus_grill-38_special_bitter	3.8
"(512) ALT" 512_brewing_company-512_alt	6
"(512) Bruin"	

Switching back to the statistics tab, you may now see 2 – 3 reads per second in the Read Ops chart:



Also, in the above screenshot, notice that the data size is different from the disk size. This is because there is some additional, extra information on the files on disk which are not necessary. This could be some updates or deletes to the append-only files on disk.
It is possible to run compaction on the index files to make the disk size match the data size.

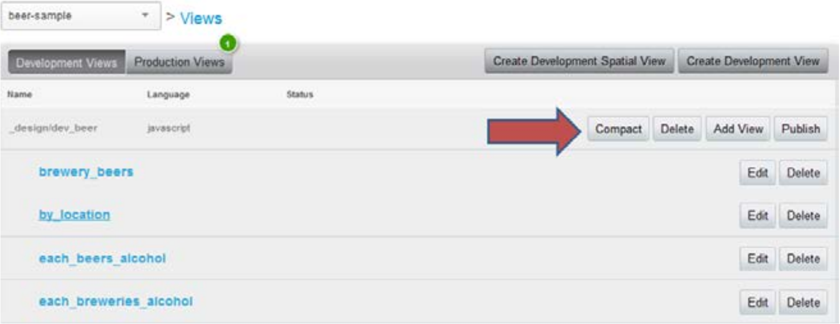
Switch to the other tab and click Views:


Couchbase - 6 Node Cluster Enterprise Edition
Documentation • Support • About • Sign Out
Overview
Server Nodes
Data Buckets
Query
Indexes
XDCR
Security
Log
Settings
Global Indexes
Views
Full Text
beer-sample
> Views >
_design/dev_beer/_views/each_beers_alcohol
manayunk_brewery_and_restaurant-brilliant_barstool
Preview a Random Document
Edit Document

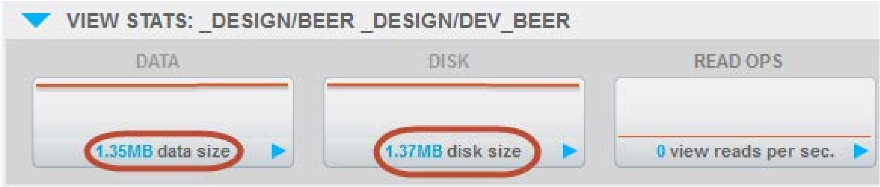
```
{
  "name": "Brilliant Barstool",
  "abv": 5.1,
  "ibu": 0,
  "srm": 0,
  "upc": 0,
  "type": "beer",
  "brewery_id": "manayunk_brewery_and_restaurant",
  "updated": "2010-07-22 20:00:20",
  "description": "A true English session ale brewed in the pale ale
style using ye olde techniques of yesteryear. Specially crafted to
```

```
{
  "id": "manayunk_brewery_and_restaurant-
brilliant_barstool",
  "rev": "1-1463162670f6000100000000002000006",
  "expiration": 0,
  "flags": 33554438
}
```

Click Compact:



Switch back to the charts tab and you should see a minor change where the disk size matches the data size more closely. In a production environment the effect of compacting an index can be more profound:



Note that compacting the files for a view is a different process than compacting data files for the original JSON data!

Investigating settings and details for Design Documents and Views:

All the views within a single design document are updated when the update to a single view is triggered. For example, a design document with four views will update all four views simultaneously when just one of these views is updated.

Updates can be triggered in two ways:

- At the point of access or query by using the stale parameter.

- Automatically by Couchbase Server based on the number of updated documents, or the period since the last update. Automatic updates can be controlled either globally, or individually on each design document.

Views are updated incrementally. The first time the view is accessed, all the documents within the bucket are processed through the map/reduce functions. Each new access to the view only processes the documents that have been added, updated, or deleted, since the last time the view index was updated.

Because of the incremental nature of the view update process, information is only ever appended to the index stored on disk. This helps ensure that the index is updated efficiently. Compaction (including auto-compaction) will optimize the index size on disk and optimize the index structure. An optimized index is more efficient to update and query.

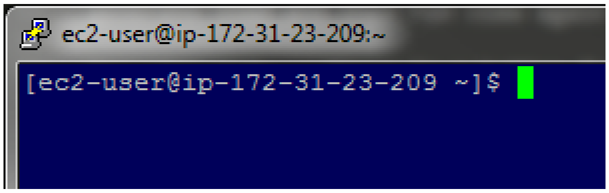
The views engine creates an index is for each design document; this index contains the results for all the views within that design document.

The index information stored on disk consists of the combination of both the key and value information defined within your view. The key and value data is stored in the index so that the information can be returned as quickly as possible, and so that views that include a reduce function can return the reduced information by extracting that data from the index.

Because the value and key information from the defined map function are stored in the index, the overall size of the index can be larger than the stored data if the emitted key/value information is larger than the original source document data.

View indexes are stored, accessed, and updated, entirely independently of the document updating system. This means that index updates and retrieval is not dependent on having documents in memory to build the index information.

With that overview of Views and Indexes, let's go to the 1st node (Blue VM) and run some commands to further explore the Indexing features:



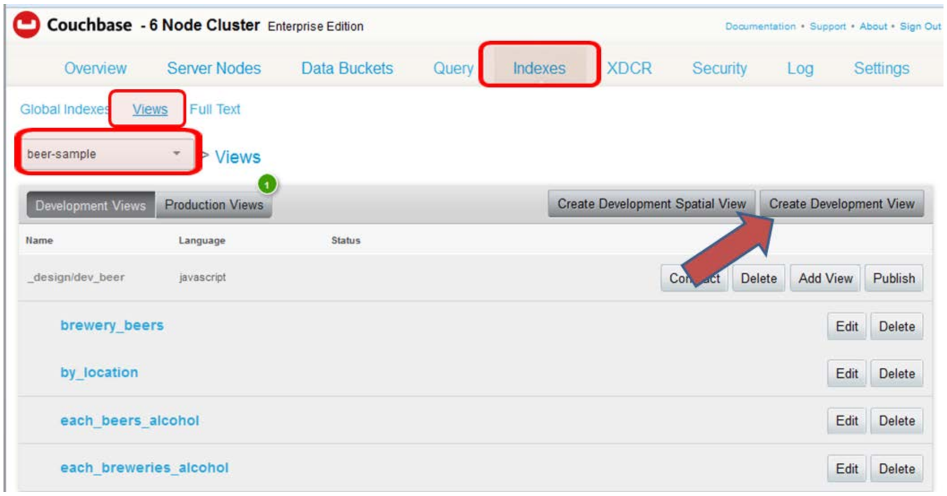
```
[ec2-user@ip-172-31-23-209 ~]$
```

Run the following command to take a look at the index files:

```
[ec2-user@ip-172-31-23-209 ~]$ sudo ls -lh
/opt/couchbase/var/lib/couchbase/index/@indexes/beer-sample
total 832K
-rw-rw----. 1 couchbase couchbase 358K Jul 21 21:08 main_90988cc003706389663f383fe0aaf9c9.view.2
drwxrwx---. 2 couchbase couchbase    6 Jul 21 19:41 tmp_90988cc003706389663f383fe0aaf9c9_main
```

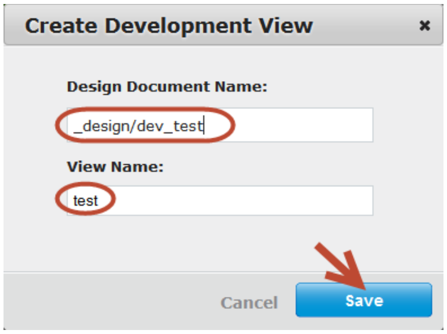
Note that there is only 1 file in this folder b/c we currently only have 1 Design Document (with 4 views) for the beer-sample bucket.

Switch back to one of the tabs with the Web UI and click on Indexes tab, then Views, then choose 'beer-sample' and click on "Create Development View"



Name	Language	Status
_design/dev_beer	javascript	
brewery_beers		
by_location		
each_beers_alcohol		
each_breweries_alcohol		

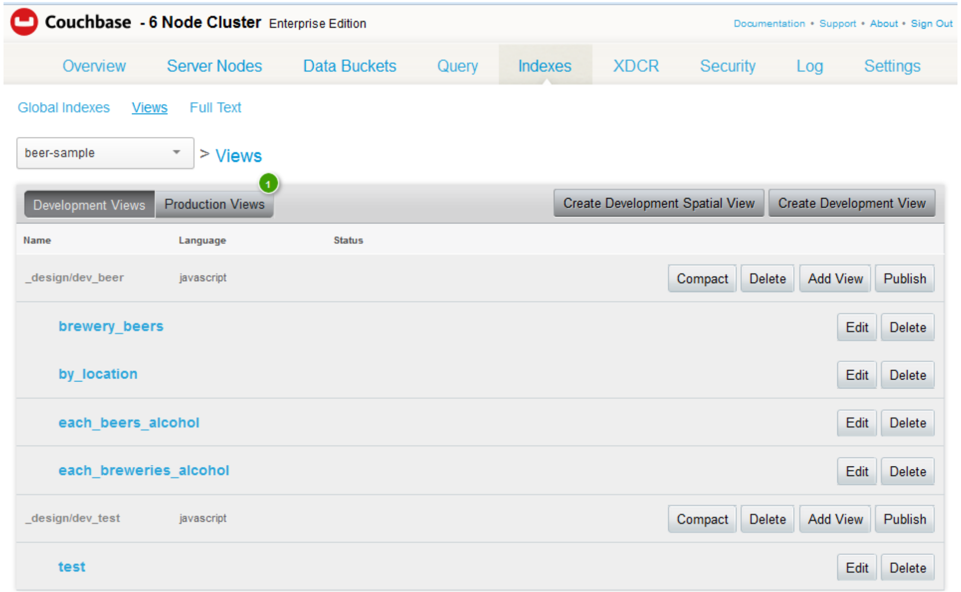
Name the design doc "_design/dev_test" and name the view 'test' and then click "Save":



Design Document Name: _design/dev_test

View Name: test

The new view will appear in the Web UI:



Name	Language	Status
_design/dev_beer	javascript	
brewery_beers		
by_location		
each_beers_alcohol		
each_breweries_alcohol		
_design/dev_test	javascript	
test		

Switch back to the 1st node's shell window and verify that there are now two files under the index folder for beer-sample:

```
 [ec2-user@ip-172-31-23-209 ~]$ sudo ls -lh
/opt/couchbase/var/lib/couchbase/index/@indexes/beer-sample
total 372K
-rw-rw----. 1 couchbase couchbase  11K Jul 21 21:40 main_193eb782fe2e7e39d621c264a36508f5.view.1
-rw-rw----. 1 couchbase couchbase 358K Jul 21 21:08 main_90988cc003706389663f383fe0aaf9c9.view.2
drwxrwx---. 2 couchbase couchbase    6 Jul 21 19:41 tmp_90988cc003706389663f383fe0aaf9c9_main
```

Next we will download a utility

```
[root@ip-172-31-23-209 ~]$ sudo -i

[root@ip-172-31-23-209 ~]$ yum install perl-LWP-Protocol-https

[root@ip-172-31-23-209 ~]$ which GET
/bin/GET
```

It is also possible to query a view via the REST API:

```
[ec2-user@ip-172-31-23-209 ~]$ GET http://localhost:8092/beer-
sample/_design/dev_beer/_view/each_beers_alcohol?limit=10

{"total_rows":5891,"rows":[
{"id":"sullivan_s_black_forest_brew_haus_grill-38_special_bitter","key":".38 Special
Bitter","value":3.8},
{"id":"512_brewing_company-512_alt","key":"(512) ALT","value":6},
{"id":"512_brewing_company-512_bruin","key":"(512) Bruin","value":7.6},
{"id":"512_brewing_company-512_ipa","key":"(512) IPA","value":7},
{"id":"512_brewing_company-512_pale","key":"(512) Pale","value":5.8},
{"id":"512_brewing_company-512_pecan_porter","key":"(512) Pecan Porter","value":6.8},
{"id":"512_brewing_company-512_whiskey_barrel_aged_double_pecan_porter","key":"(512) Whiskey
Barrel Aged Double Pecan Porter","value":8.2},
{"id":"512_brewing_company-512_wit","key":"(512) Wit","value":5.2},
{"id":"big_ridge_brewing-17_cream_ale","key":"#17 Cream Ale","value":0},
{"id":"minocqua_brewing_company-40_golden_lager","key":"#40 Golden Lager","value":0}
]
}
```

Further information about how to use the REST API to query a view for a specific key or doing things like grouping or setting reduce to true can be found here:
http://docs.couchbase.com/admin/admin/rest-intro.html

It is also possible to retrieve a design document via the REST API. For example, this command will retrieve all of the 4 dev views in the beer-sample bucket:

```
[ec2-user@ip-172-31-23-209 ~]$ curl -X GET -H 'Content-Type:
application/json' 'http://localhost:8092/beer-
sample/_design/dev_beer'
```

Or this may be added to the end of the statement

```
| python –mjson.tool
```

<note the results below have been formatted using a JSON formatter for viewability>

```
{
   "language":"javascript",
   "views":{
      "brewery_beers":{
         "map":"function(doc, meta) {\n  switch(doc.type) {\n  case
\"brewery\":\n    emit([meta.id]);\n    break;\n  case \"beer\":\n    if
(doc.brewery_id) {\n      emit([doc.brewery_id, meta.id]);\n    }\n
break;\n  }\n}\n"
      },
      "by_location":{
         "map":"function (doc, meta) {\n  if (doc.country, doc.state,
doc.city) {\n    emit([doc.country, doc.state, doc.city], 1);\n  } else if
(doc.country, doc.state) {\n    emit([doc.country, doc.state], 1);\n  } else
if (doc.country) {\n    emit([doc.country], 1);\n  }\n}",
         "reduce":"_count"
      },
      "each_beers_alcohol":{
         "map":"function (doc, meta) {\n  if (doc.type == \"beer\" &&
doc.brewery_id && doc.name) {\n  emit(doc.name, doc.abv);\n  }\n}"
      },
      "each_brewerys_alcohol":{
         "map":"function (doc, meta) {\n  if (doc.type == \"beer\" &&
doc.brewery_id) {\n  emit(doc.brewery_id, doc.abv);\n  }\n}\n\n",
         "reduce":"_stats"
      }
   }
}
```

Recall from the introduction to this section that views can be updated automatically based on the number of updated documents, or the period since the last update.

You configure automated update through two parameters, the update time interval in seconds and the number of document changes that occur before the views engine updates an index. These two parameters are updateInterval and updateMinChanges :

updateInterval: the time interval in milliseconds, default is 5000 milliseconds. At every updateInterval the views engine checks if the number of document mutations on disk is greater than updateMinChanges. If true, it triggers view update. The documents stored on disk potentially lag documents that are in-memory for tens of seconds.
updateMinChanges: the number of document changes that occur before re-indexing occurs, default is 5000 changes.

We can retrieve both of these values via the REST API:

```
[ec2-user@ip-172-31-23-209 ~]$ GET
http://Administrator:couchbase@localhost:8091/settings/viewUpdateDaemon

{"updateInterval":5000,"updateMinChanges":5000,"replicaUpdateMinChange
s":5000}
```

Let's change the updateInterval to 7 seconds:

```
 [ec2-user@ip-172-31-23-209 ~]$ POST
http://Administrator:couchbase@localhost:8091/settings/viewUpdateDaemo
n<hit enter>

Please enter content (application/x-www-form-urlencoded) to be POSTed:
updateInterval=7000<carefully hit 'CTRL + D' two times>

{"updateInterval":7000,"updateMinChanges":5000,"replicaUpdateMinChange
s":5000}
```

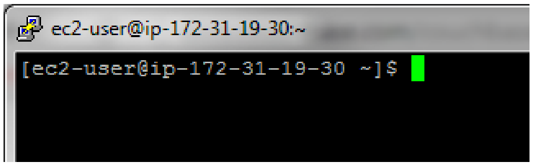
The results from the POST command will show that the updateInterval has now been set to 7000.

Note that this change takes effect across all of the design documents and on all nodes of the cluster. It is also possible to make this change on specific design documents as outlined in the following link:
http://docs.couchbase.com/admin/admin/Views/views-operation.html

Generating load for the beer-bucket to trigger the indexer to run:

In the final section for this lab, we will generate load against the beer-sample bucket and expect the indexer to get triggered every 7 seconds or as every 5,000 items are added.

Switch to your App Server:



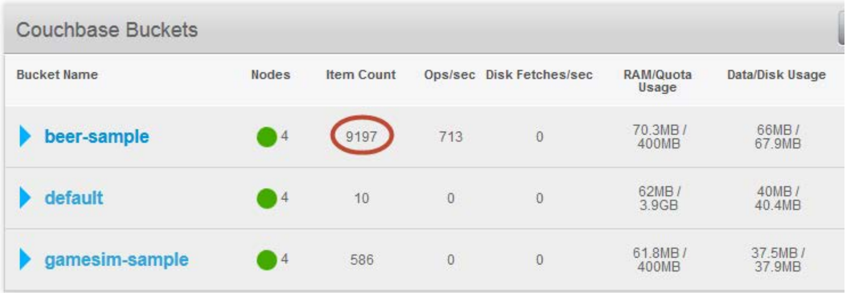
```
ec2-user@ip-172-31-19-30:~
[ec2-user@ip-172-31-19-30 ~]$
```

Run the following cbworkloadgen command to generate some test data in the beer-sample bucket. You can provide public hostname of the 1st node within this command:

```
[ec2-user@ip-172-31-19-30 ~]$ /opt/couchbase/bin/cbworkloadgen -n ec2-
54-85-43-x.compute-1.amazonaws.com:8091 -u Administrator -p couchbase
-b beer-sample -i 250000 -r 1 -s 10
  [###                    ] 14.5% (29000/estimated 250000 msgs)
```

The above command should take around 30 seconds to run.

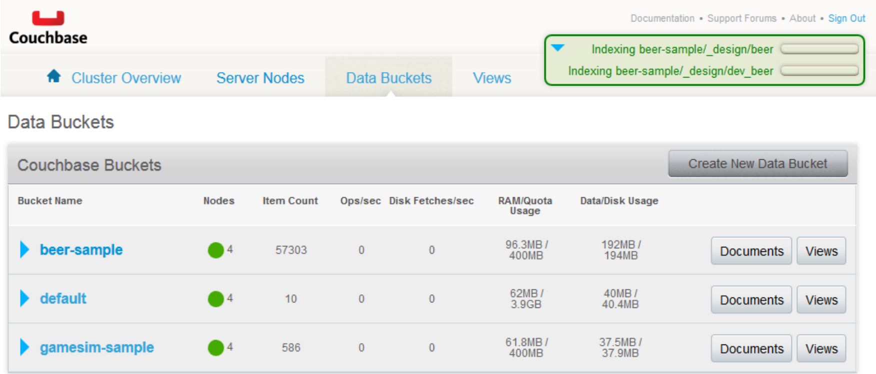
While it is running, quickly switch over to the Web UI and you will see 250,000 additional items being added in the beer-sample bucket for a final total of 257,303. So the item count will be growing for about 30 seconds till it stabilizes at the final total.



Couchbase Buckets

Bucket Name	Nodes	Item Count	Ops/sec	Disk Fetches/sec	RAM/Quota Usage	Data/Disk Usage
beer-sample	4	9197	713	0	70.3MB / 400MB	66MB / 67.9MB
default	4	10	0	0	62MB / 3.9GB	40MB / 40.4MB
gamesim-sample	4	586	0	0	61.8MB / 400MB	37.5MB / 37.9MB

Furthermore, every 7 seconds, you should see a green message at the top right of the webpage mentioning that the beer-sample bucket is being indexed:



Indexing beer-sample/_design/beer

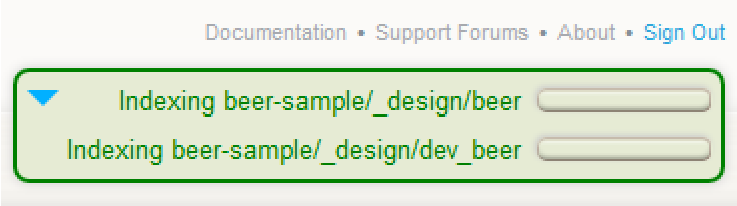
Indexing beer-sample/_design/dev_beer

Data Buckets

Couchbase Buckets

Bucket Name	Nodes	Item Count	Ops/sec	Disk Fetches/sec	RAM/Quota Usage	Data/Disk Usage
beer-sample	● 4	57303	0	0	96.3MB / 400MB	192MB / 194MB
default	● 4	10	0	0	62MB / 3.9GB	40MB / 40.4MB
gamesim-sample	● 4	586	0	0	61.8MB / 400MB	37.5MB / 37.9MB

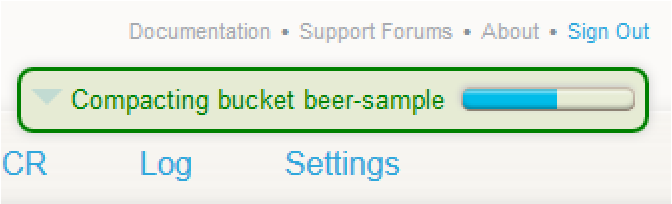
The message will look like this:



Indexing beer-sample/_design/beer

Indexing beer-sample/_design/dev_beer

You might also see the beer-sample bucket being compacted. Note, this is not the index or views being compacted, but the actual data bucket itself:



Compacting bucket beer-sample

This concludes Lab #5.