CS300 Couchbase NoSQL Server Administration

Lab 2 Exercise Manual

Release: 4.5

Revised: Sept 1st, 2016

Lab #2: Installation and configuration of a client app server

Objective: This 1-hour lab will walk you through connecting to and configuring a new Virtual Machine in Amazon's cloud to act as an application client that we can simulate load from using various load generation tools like cbworkloadgen and pillowfight. In the lab you will submit some reads and writes against the 1-node Couchbase cluster and learn how to verify that the cluster is running and accepting reads + writes.

Warning: Do not copy + paste commands from this lab into your PuTTY/Terminal session. Some commands, especially multi-line commands will not paste properly and the ASCII symbols from the PDF will not appear the same in the SSH session. A multi-line command will break into 2 lines when you copy it as the PDF will insert a /n character after the first line. This will cause the line to be split incorrectly when you paste it into the terminal window. Instead, please type each command individually into the SSH session!

Please send any comments or corrections in this lab or future labs to Couchbase Learning Services at cls@couchbase.com

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

- Run cbworkloadgen from the existing 1-node Couchbase cluster
- Connect to a new VM in the same availability zone as the first Couchbase node and prepare it for simulating read/write load via various client apps (cbworkloadgen, telnet, cbc, pillowfight)
- Using cbworkloadgen, read and write data to the 1-node cluster
- Test the Memcached text protocol using telnet (stats, put a key, get a key)
- Learn how to use the Rest API
- Run pillowfight to read/write date to the 1-node cluster
- Use the cbc command to create, read and delete a key in the cluster

Using cbworkloadgen Tool:

cbworkloadgen is a tool that generates random data and performs reads/writes for Couchbase Server. This tool provides basic testing functionality but is not designed for real-world performance or stress testing. It has options for tuning the ratio of read (get) vs. write (set) operations, the number and size of the documents inserted and the number of concurrent worker threads.

In Linux, the tool is located here:

```
/opt/couchbase/bin/cbworkloadgen
```

Let's test the installation of Couchbase Server by using cbworkloadgen to insert some random data into the cluster.

Switch to the PuTTY or Blue Terminal window for the 1st node(couchbase01) and…

Print the help menu for the command formatting for this tool:

```
[ec2-user@Couchbase01 ~]$ cbworkloadgen --help
Usage: cbworkloadgen [options]

Generate workload to destination.

Examples:
  cbworkloadgen -n localhost:8091
  cbworkloadgen -n 10.3.121.192:8091 -r .9 -i 100000 \
         -s 100 -b my-other-bucket --threads=10


Options:
  -h, --help            show this help message and exit
  -r .95, --ratio-sets=.95
                        set/get operation ratio
  -n 127.0.0.1:8091, --node=127.0.0.1:8091
                        node's ns_server ip:port
  -b default, --bucket=default
                        insert data to a different bucket other than default
  --ssl                 Transfer data with SSL enabled
  -i 10000, --max-items=10000
                        number of items to be inserted
  -s 10, --size=10      minimum value size
  --prefix=pymc         prefix to use for memcached keys or json ids
  -j, --json            insert json data
  -l, --loop            loop forever until interrupted by users
  -u USERNAME, --username=USERNAME
                        REST username for cluster or server node
  -p PASSWORD, --password=PASSWORD
                        REST password for cluster or server node
  -t 1, --threads=1     number of concurrent workers
  -v, --verbose         verbose logging; more -v's provide more verbosity
  --low-compression     generate document data that is difficult to compress
```

Run cbworkloadgen with no options:

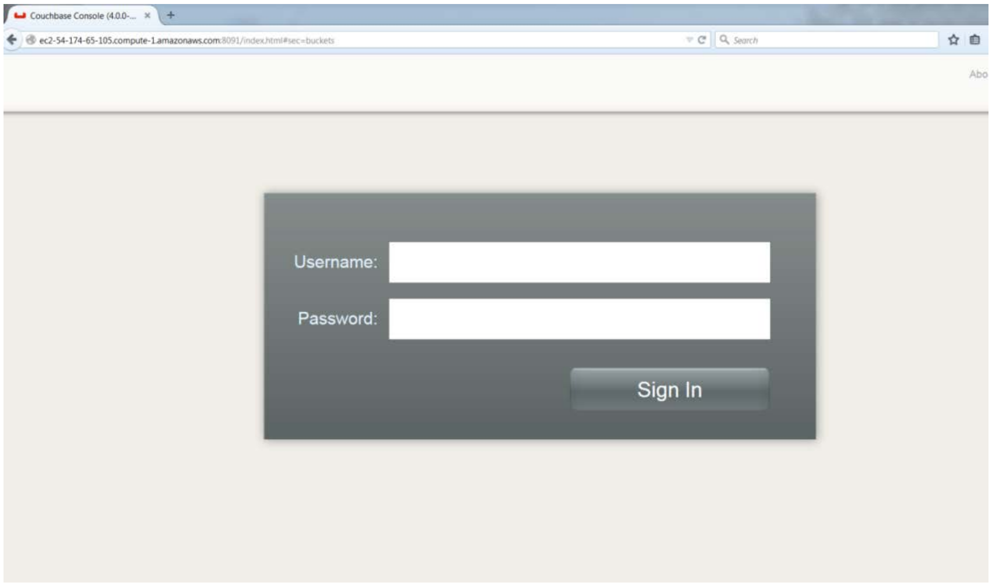
```
 [ec2-user@Couchbase01 ~]$ cbworkloadgen
[####################] 100.0% (10527/estimated 10526 msgs)
bucket: default, msgs transferred...
       :                total |       last |    per sec
 byte  :               105270 |     105270 |    214409.7
done
```

Re-run cbworkloadgen with localhost, username & password:

```
[ec2-user@Couchbase01 ~]$ cbworkloadgen -n localhost:8091 -u
Administrator -p couchbase
  [####################] 100.0% (10527/estimated 10526 msgs)
bucket: default, msgs transferred...
       :                total |       last |    per sec
 byte  :               105270 |     105270 |    217014.8
done
```

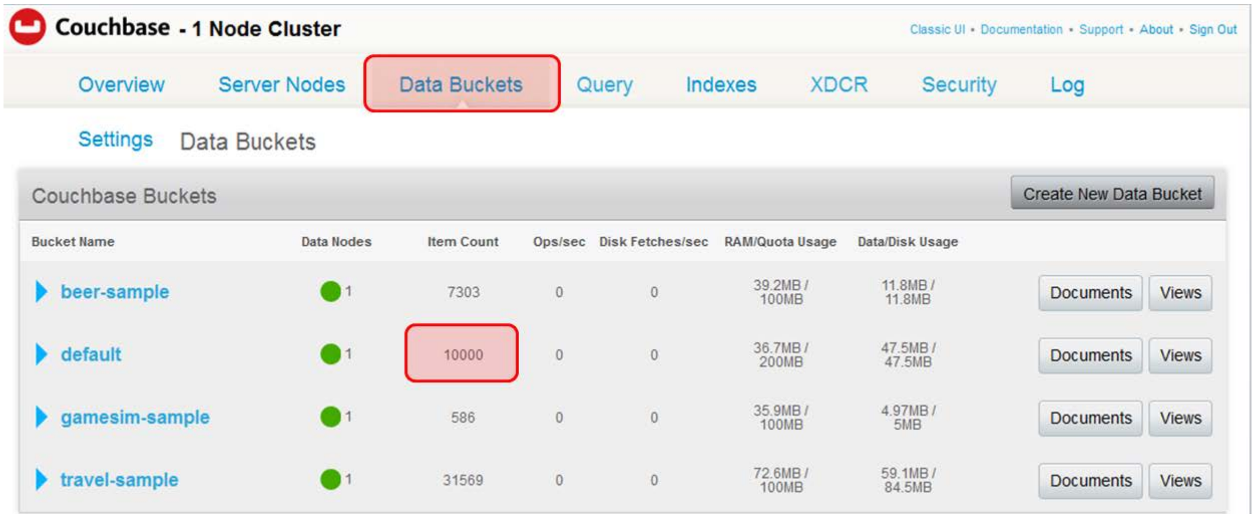
The default settings in cbworkloadgen will insert 10,000 items into Couchbase.

Switch to your browser window and reconnect to the Couchbase Web UI Console. You may need to log back in:



Couchbase Console (4.0.0-...
ec2-54-174-65-105.compute-1.amazonaws.com:8091/index.html#sec=buckets
Search
Abo
Username:
Password:
Sign In

Click on the 'Data Buckets' link at the top and you'll see that there are now 10,000 items in the default bucket:



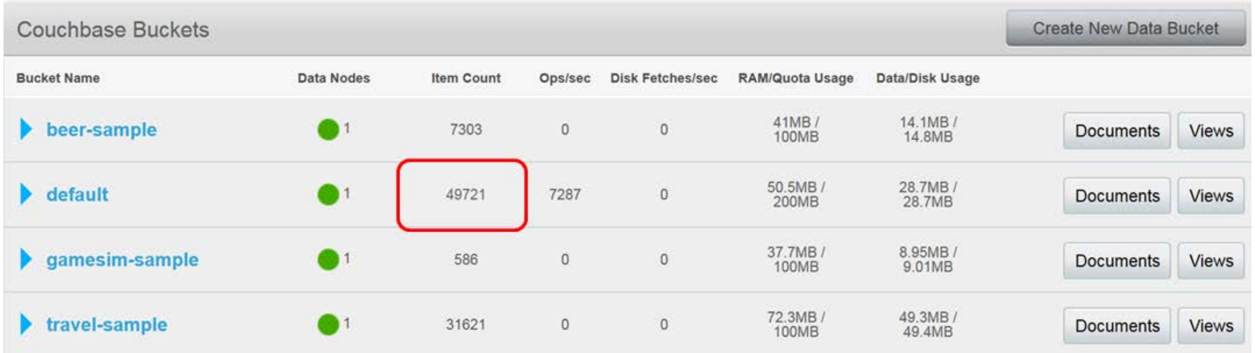
Bucket Name	Data Nodes	Item Count	Ops/sec	Disk Fetches/sec	RAM/Quota Usage	Data/Disk Usage		
beer-sample	● 1	7303	0	0	39.2MB / 100MB	11.8MB / 11.8MB	Documents	Views
default	● 1	10000	0	0	36.7MB / 200MB	47.5MB / 47.5MB	Documents	Views
gamesim-sample	● 1	586	0	0	35.9MB / 100MB	4.97MB / 5MB	Documents	Views
travel-sample	● 1	31569	0	0	72.6MB / 100MB	59.1MB / 84.5MB	Documents	Views

Return to the PuTTY or Terminal window and…

Run cbworkloadgen to insert 500,000 items of size 10 bytes with 50% of the workload set to writes:

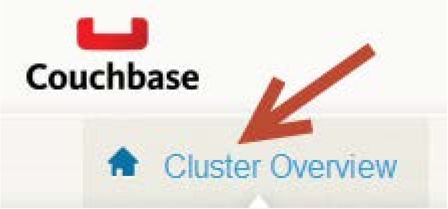
```
[ec2-user@Couchbase01 ~]$ cbworkloadgen -n localhost:8091 -u
Administrator -p couchbase -i 500000 -r .5 -s 10
[####################] 100.0% (999999/estimated 1000000 msgs)
bucket: default, msgs transferred...
       :                total |       last |    per sec
 byte  :              9999990 |    9999990 |   259225.6
done
```

The above command will take about 1-2 minutes to run. While this workload is running, quickly continue with the next few steps. First refresh the Couchbase Web UI and you should see more items added to the default bucket:

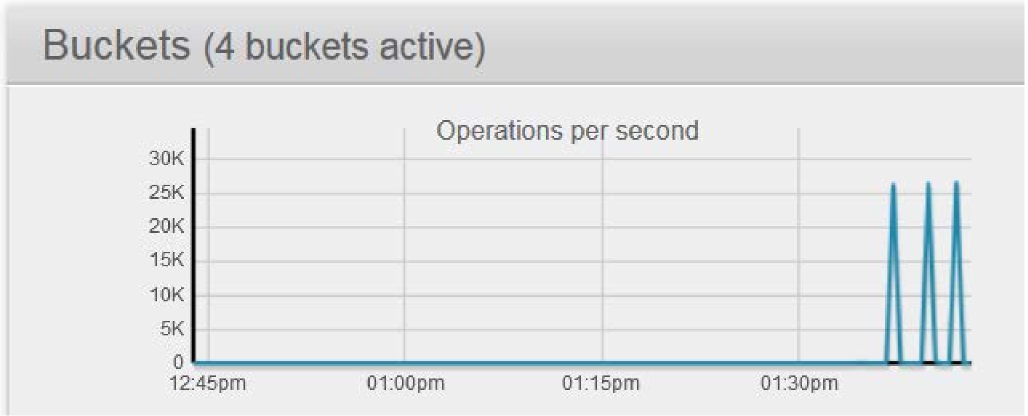


Bucket Name	Data Nodes	Item Count	Ops/sec	Disk Fetches/sec	RAM/Quota Usage	Data/Disk Usage		
beer-sample	● 1	7303	0	0	41MB / 100MB	14.1MB / 14.8MB	Documents	Views
default	● 1	49721	7287	0	50.5MB / 200MB	28.7MB / 28.7MB	Documents	Views
gamesim-sample	● 1	586	0	0	37.7MB / 100MB	8.95MB / 9.01MB	Documents	Views
travel-sample	● 1	31621	0	0	72.3MB / 100MB	49.3MB / 49.4MB	Documents	Views

Click on 'Cluster Overview' on the top left of the screen:

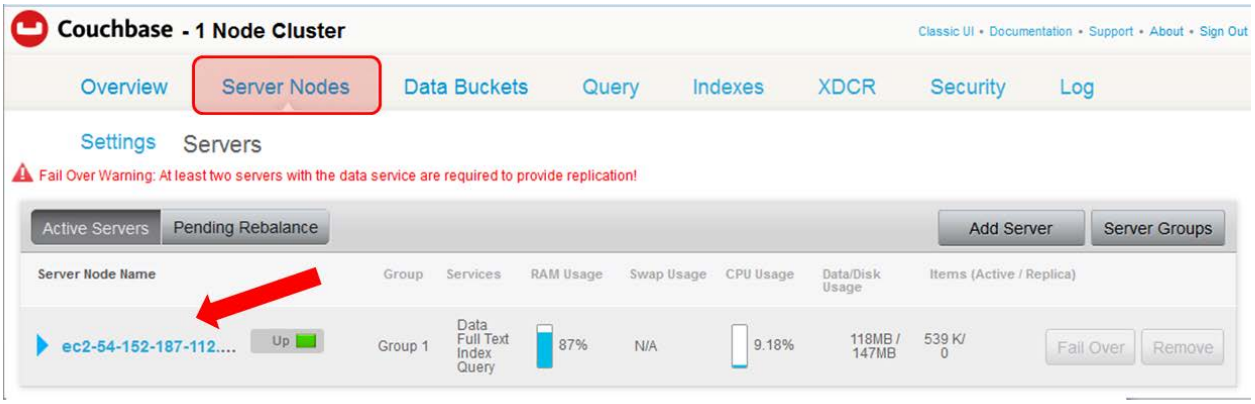


Scroll down until you see the Ops per second graph:



Notice that this one node is servicing about 20k-25k, operations per second. In your specific cloud environment, the range of ops per second could be between 15k – 25k ops per second.

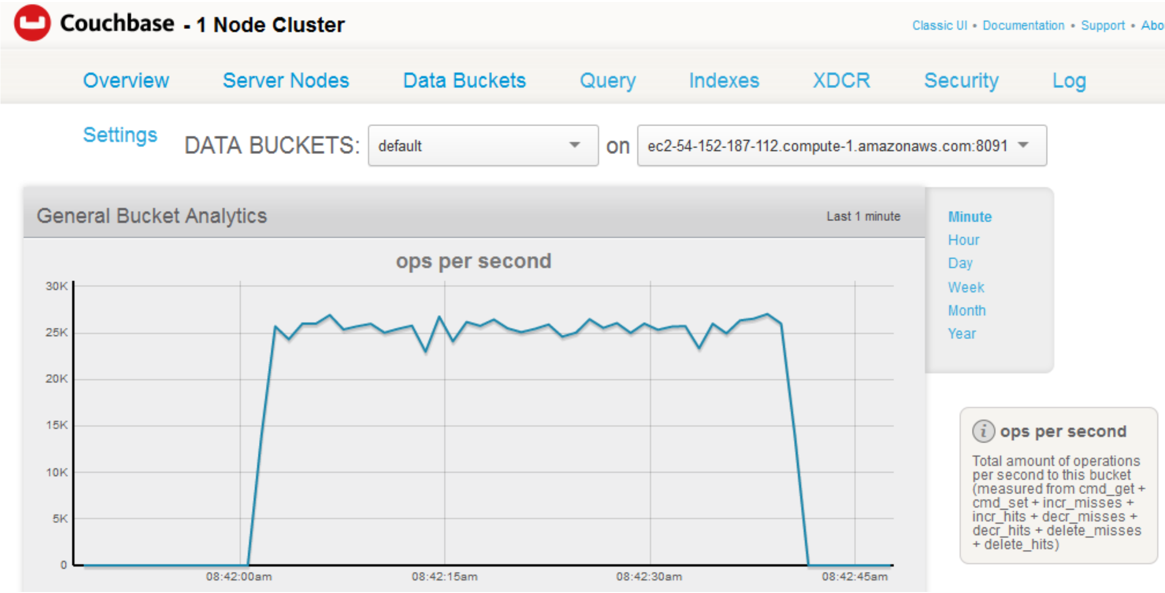
You can get more detailed performance graphs, by clicking on "Server Nodes" at the top and then clicking on the specific server's IP Node Name in the page:



Fail Over Warning: At least two servers with the data service are required to provide replication!

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	87%	N/A	9.18%	118MB / 147MB	539 K/ 0

The resulting page will look like this:



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)

We will explore these graphs in a future performance lab in depth.

For now, return back to the cmd-line and check if the tool has finished running:

```
  [####################] 100.0% (999999/estimated 1000000 msgs)
bucket: default, msgs transferred...
       :                total |       last |    per sec
 byte  :              9999990 |    9999990 |    253711.0
```

Notice that the tool performed 253711.0 bytes of I/O per second into the default bucket.(your numbers will vary based on Vcpu's and memory of the VM you are working on) It performed a total of 1,000,000 (1 million) operations , which makes sense… since we wanted to insert 500,000 new items and wanted the inserts (sets) to be 50% of the overall ratio.

Connect to the application client:

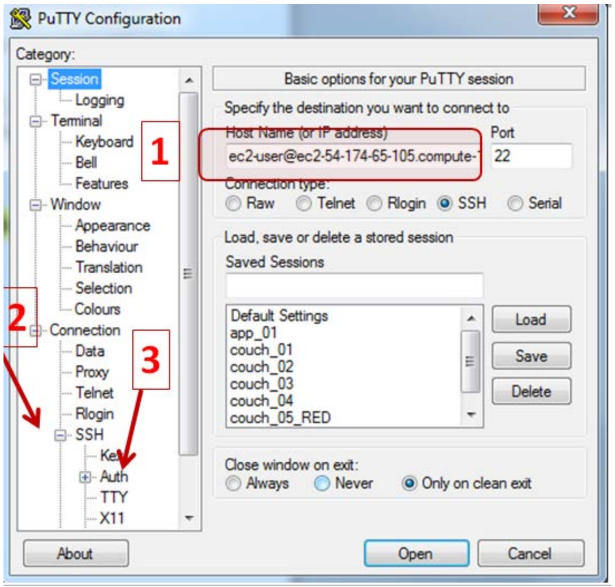
Now that we have verified that Couchbase Server is working and accepting fresh writes from a local client, next we will set up and configure a new client application server.

The application client server you have been assigned has the following characteristics:

Amazon Instance Type: **t2.medium**
ECUs: **3**
vCPU: **2**
Memory: **3.75 GiB**
Storage: **10 GB**
Network performance: **moderate**
CloudWatch Monitoring: **disabled**
Tenancy: **Shared tenancy** (multi-tenant hardware)
Cost: **$0.05 per hour**

Launch PuTTY and connect to the Application Server.
After starting PuTTY, enter the Amazon DNS name of your Application Server VM into PuTTY. You can get this DNS name from the `Cluster-IPs` spreadsheet that the instructor gave you along with this lab. The connection type will be SSH and the port will be 22.

Type "ec2-user@<public hostname>" with the public hostname that the instructor gave you for the App Server into PuTTY and then **click on the + next to SSH** to expand its options and finally **select Auth**:



Specify the destination you want to connect to

Host Name (or IP address): ec2-user@ec2-54-174-65-105.compute-1

Port: 22

Connection type: ○ Raw ○ Telnet ○ Rlogin ● SSH ○ Serial

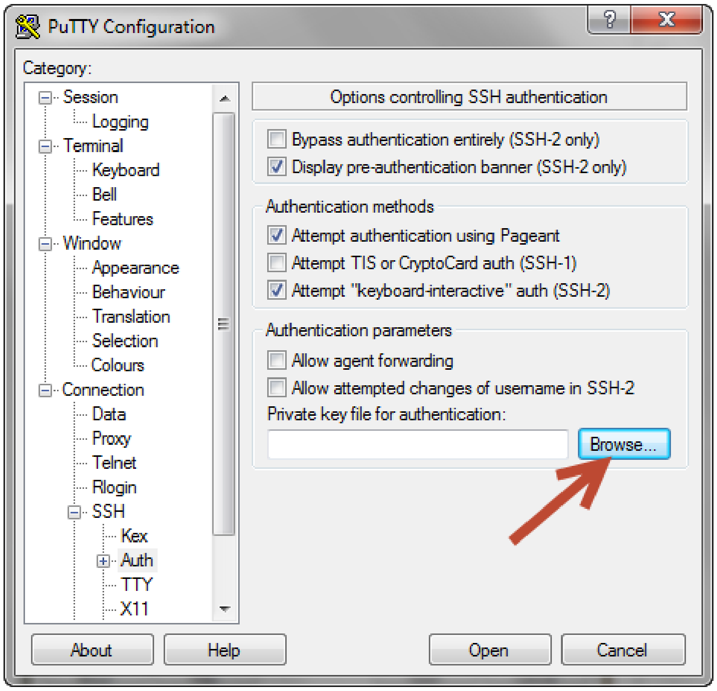
Load, save or delete a stored session

Saved Sessions

Default Settings
app_01
couch_01
couch_02
couch_03
couch_04
couch_05_RED

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

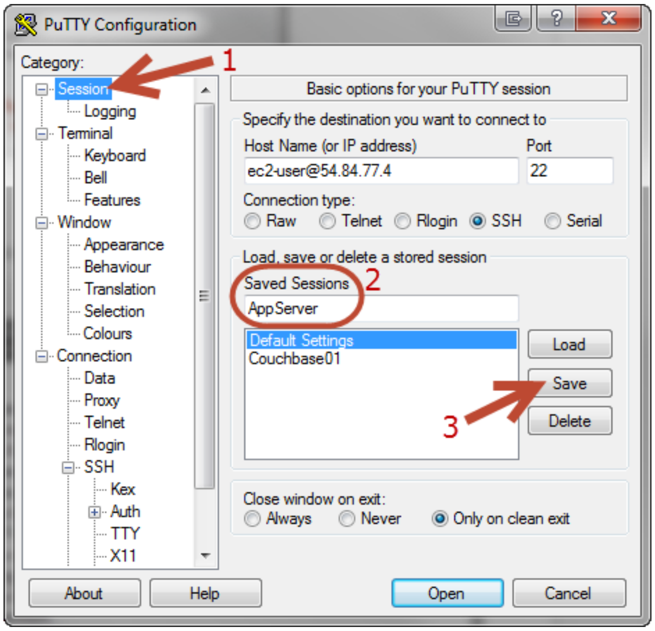
☑ Attempt authentication using Pageant
□ Attempt TIS or CryptoCard auth (SSH-1)
☑ Attempt "keyboard-interactive" auth (SSH-2)

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.
Next, **click on Session** and type to **save the session as "AppServer"**. Then **click on Save**.



Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address): ec2-user@54.84.77.4

Port: 22

Connection type:
○ Raw ○ Telnet ○ Rlogin ● SSH ○ Serial

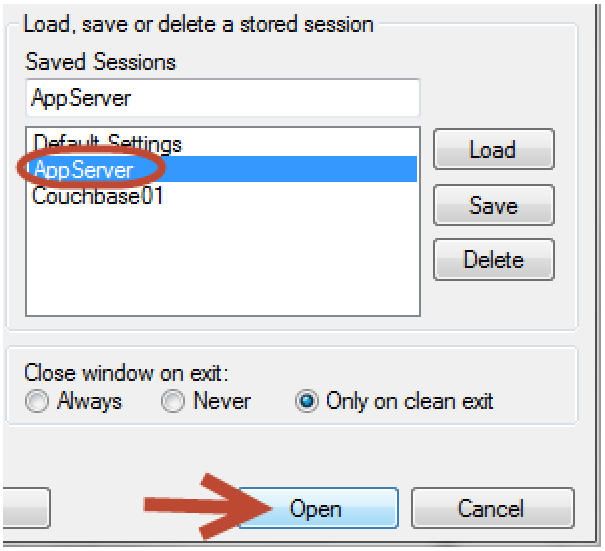
Load, save or delete a stored session

Saved Sessions: AppServer

Default Settings
Couchbase01

Close window on exit:
○ Always ○ Never ● Only on clean exit

Now highlight **AppServer** and click **Open** to connect to this VM:



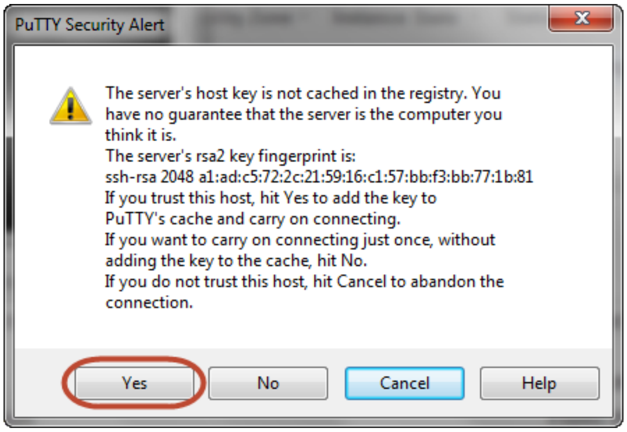
Load, save or delete a stored session

Saved Sessions: AppServer

Default Settings
AppServer
Couchbase01

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.



PuTTY Security Alert

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.

Configure the client server and install Couchbase on it:

Next, we will quickly run through some steps to configure this server by turning off the firewall, etc and then install Couchbase Server 4.5.0 on it so we can easily get the cbworkloadgen tool (along with some other tools). We don't actually need Couchbase running on this server, so we will stop the Couchbase service immediately after the installation.

Become root

```
# sudo –i
```

Set the hostname to AppServer

```
# hostname AppServer
```

Turn off the linux firewall: (if necessary)

```
# systemctl stop firewalld
```

Install wget

```
# yum install wget

# exit
```

Close the putty window and open a new one to verify successful hostname change.

Download Couchbase 4.5.0 EE (*do not copy + paste this command!*):

```
[ec2-user@AppServer ~]$  wget
http://packages.couchbase.com/releases/4.5.0/couchbase-server-
enterprise-4.5.0-centos7.x86_64.rpm
--2016-04-26 13:31:47--  http://packages.couchbase.com/releases/4.5.0-DP1/couchbase-server-
enterprise-4.5.0-DP1-centos7.x86_64.rpm
Resolving packages.couchbase.com (packages.couchbase.com)... 54.231.16.240
Connecting to packages.couchbase.com (packages.couchbase.com)|54.231.16.240|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 100473860 (96M) [application/x-rpm]
Saving to: 'couchbase-server-enterprise-4.5.0-DP1-centos7.x86_64.rpm'

100%[============================================================>] 100,473,860 58.0MB/s   in 1.7s
2016-04-26 13:31:48 (58.0 MB/s) - 'couchbase-server-enterprise-4.5.0-DP1-centos7.x86_64.rpm'
saved [100473860/100473860]
```

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

```
[ec2-user@ AppServer ~]$ sudo rpm --install couchbase-server-
enterprise-4.5.0-centos7.x86_64.rpm

Warning: Transparent hugepages looks to be active and should not be.
Please look at http://bit.ly/1ZAcLjD as for how to PERMANENTLY alter this setting.
Warning: Swappiness is not set to 0.
Please look at http://bit.ly/1k2CtNn as for how to PERMANENTLY alter this setting.
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

Reloading systemd:                                         [  OK  ]
Starting couchbase-server (via systemctl):                 [  OK  ]

You have successfully installed Couchbase Server.
Please browse to http://AppServer: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: 11211, 11210, 11209, 4369,
8091, 8092, 8093, 9100 to 9105, 9998, 18091, 18092, 11214, 11215 and
from 21100 to 21299.

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

Note: warning about Transparent Hugepages and Swappiness. We are going to turn Couchbase server off so it will not be needed on this node however you should remember this if you see it while installing on a production node.

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

```
[ec2-user@ AppServer ~]$ sudo /etc/init.d/couchbase-server status
couchbase-server is running
```

Since we aren't planning on using this node as an actual Couchbase Server cluster node, go ahead and stop Couchbase on it:

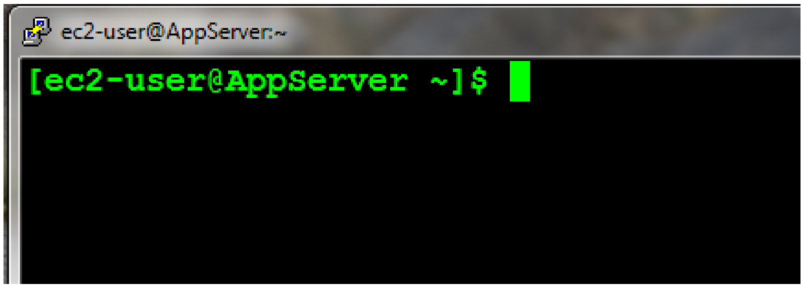
```
[ec2-user@ AppServer ~]$ sudo /etc/init.d/couchbase-server stop
Stopping couchbase-server (via systemctl):                 [  OK  ]

[ec2-user@ AppServer ~]$ sudo /etc/init.d/couchbase-server status
couchbase-server is not running
```

Run cbworkloadgen from App Client:

Next, we'll attempt running the cbworkloadgen from the new App Client.

You should currently be logged into the App Client PuTTY/Terminal shell.



```
[ec2-user@AppServer ~]$
```

Enter the cbworkloadgen directory:

```
[ec2-user@ AppServer ~]$ cd /opt/couchbase/bin
```

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

```
[ec2-user@ AppServer bin]$ ./cbworkloadgen -n ec2-54-85-43-x.compute-1.amazonaws.com:8091 -u Administrator -p couchbase
  [####################] 100.0% (10527/estimated 10526 msgs)
bucket: default, msgs transferred...
       :                total |       last |    per sec
 byte  :               105270 |     105270 |   290846.9
```

Couchbase course materials are exclusively for use by a single participant in a hands-on training course delivered by Couchbase, Inc. or a Couchbase Authorised Training Partner, as listed at www.couchbase.com Use or distribution other than to a participant in such training event is prohibited. If you believe these course materials have been reproduced or distributed in print or electronic without permission of Couchbase, Inc. please email: training@couchbase.com

Copyright © 2014 Couchbase, Inc. All rights reserved CS300 Couchbase NoSQL Server Administration

```
done
```

note: your amount per sec will vary based on Vcpu's and Memory

Excellent! The above output means that about 10,527 operations were successfully conducted against the 1-node Couchbase cluster.

Try writing 100,000 items of size 10 bytes with 50% of the workload set to writes:

```
[ec2-user@ AppServer bin]$ ./cbworkloadgen -n ec2-54-85-43-x.compute-1.amazonaws.com:8091 -u Administrator -p couchbase -i 100000 -r .5 -s 10
```

The command should take about 10 seconds to complete with similar results to this:

```
  [####################] 100.0% (199999/estimated 200000 msgs)
bucket: default, msgs transferred...
       :                total |       last |    per sec
 byte  :              1999990 |    1999990 |   321261.4
done
```

If you remember from earlier in this lab, when we ran cbworkloadgen on the same VM as the Couchbase Server, we saw about 25,000 ops per second (your mileage might vary, depending on the dynamic cloud conditions in the Amazon datacenter). In my specific case, my client app is reporting about 321261.4 bytes of I/O per second.

Run telnet from App Client:

You can also test your Couchbase Server installation by using Telnet to connect to the server and using the Memcached text protocol. This is the simplest method for determining if Couchbase is running.

Run the commands below, from the AppServer VM and telnet into the public IP of the 1st Couchbase Server:

```
[ec2-user@ AppServer bin]$ cd ~
[ec2-user@ AppServer ~]$

[ec2-user@ AppServer ~]$ sudo yum -y install telnet
Loaded plugins: amazon-id, rhui-lb
rhui-REGION-client-config-server-7                                                              |
2.9 kB  00:00:00stats
rhui-REGION-rhel-server-releases                                                                |
3.7 kB  00:00:00
rhui-REGION-rhel-server-rh-common                                                               |
1.9 kB  00:00:00
(1/4): rhui-REGION-rhel-server-rh-common/7Server/x86_64/updateinfo                              |
11 kB  00:00:00
```

```
(2/4): rhui-REGION-rhel-server-rh-common/7Server/x86_64/primary                                |
30 kB  00:00:00
(3/4): rhui-REGION-client-config-server-7/x86_64/primary_db                                    |
4.3 kB  00:00:00
(4/4): rhui-REGION-rhel-server-releases/7Server/x86_64/primary_db                              |
12 MB  00:00:01
(1/2): rhui-REGION-rhel-server-releases/7Server/x86_64/updateinfo                              |
528 kB  00:00:00
(2/2): rhui-REGION-rhel-server-releases/7Server/x86_64/group_gz                                |
133 kB  00:00:00
rhui-REGION-rhel-server-rh-common
131/131
Resolving Dependencies
--> Running transaction check
---> Package telnet.x86_64 1:0.17-59.el7 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================================
=====================
 Package            Arch              Version                     Repository
Size
================================================================================================
=====================
Installing:
 telnet             x86_64            1:0.17-59.el7               rhui-REGION-rhel-server-
releases              63 k

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

Total download size: 63 k
Installed size: 113 k
Downloading packages:
telnet-0.17-59.el7.x86_64.rpm                                                                  |
63 kB  00:00:00
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
Warning: RPMDB altered outside of yum.
  Installing : 1:telnet-0.17-59.el7.x86_64
1/1
  Verifying  : 1:telnet-0.17-59.el7.x86_64
1/1

Installed:
  telnet.x86_64 1:0.17-59.el7

Complete!
```

From the AppClient's PuTTY/Terminal window, type in the public hostname of the 1st Couchbase VM and connect via port 11211:

```
[ec2-user@ AppServer ~]$ telnet ec2-54-208-47-x.compute-
1.amazonaws.com 11211
Trying 54.85.188.239...
Connected to 54.85.188.239.
Escape character is '^]'.
```

While in the Telnet shell, try a few commands to test the connectivity to Couchbase.

Stats is a great way to check basic health:

```
stats
STAT delete_misses 0
STAT ep_io_num_write 620000
STAT ep_tap_requeue_sleep_time 0.1
STAT ep_config_file
STAT ep_num_access_scanner_runs 0
STAT ep_vb_snapshot_total 5122
<output truncated>
END
```

Note that you don't get the full 'cbstats' command output from the telnet stats command. To get the maximum amount of details, run the Couchbase cbstats tool. We will explore 'cbstats' in depth in future labs, but you can start researching the tool here: http://docs.couchbase.com/admin/admin/CLI/cbstats-intro.html

The syntax for setting a key is: `set <key> <flags> <exptime> <bytes> [noreply] \r\n <value> \r\n`

The parameters mean:

<key> : the key of the data stored
<flags> : 32-bit unsigned integer that the server store with the data (provided by the user), and return along the data when the item is retrieved
<exptime> : expiration time in seconds, 0 mean no delay, if exptime is superior to 30 day, Memcached will use it as a UNIX timestamps for expiration
<bytes> : number of bytes in the data block
<cas unique> : unique 64-bit value of an existing entry (retrieved with gets command) to use with cas command
[noreply] : optional parameter that inform the server to not send the reply

The memcached commands can return:

STORED to indicate success
NOT_STORED indicate that the data was not stored because condition for "add" or "replace" command wasn't met, or the item is in a delete queue
EXISTS indicate that the item you are trying to store with "cas" command has been modified since last fetch
NOT_FOUND indicate that the item did not exist or has been deleted

Put a key in with 0 as the flag, expiration time of 300 seconds (5 mins) and of size of 4 bytes. Note, this is a 2-line command. First enter the 'set' line, hit enter, then type in the value part.

```
set test_key 0 300 4  <hit enter on the keyboard>
data
STORED
```

A STORED reply means the data was successfully submitted.

Retrieve the key:

```
get test_key
VALUE test_key 0 4
data
END
```

The retrieval command returns:

```
VALUE <flag> <bytes>\r\n<data>
END indicate the end of response
```

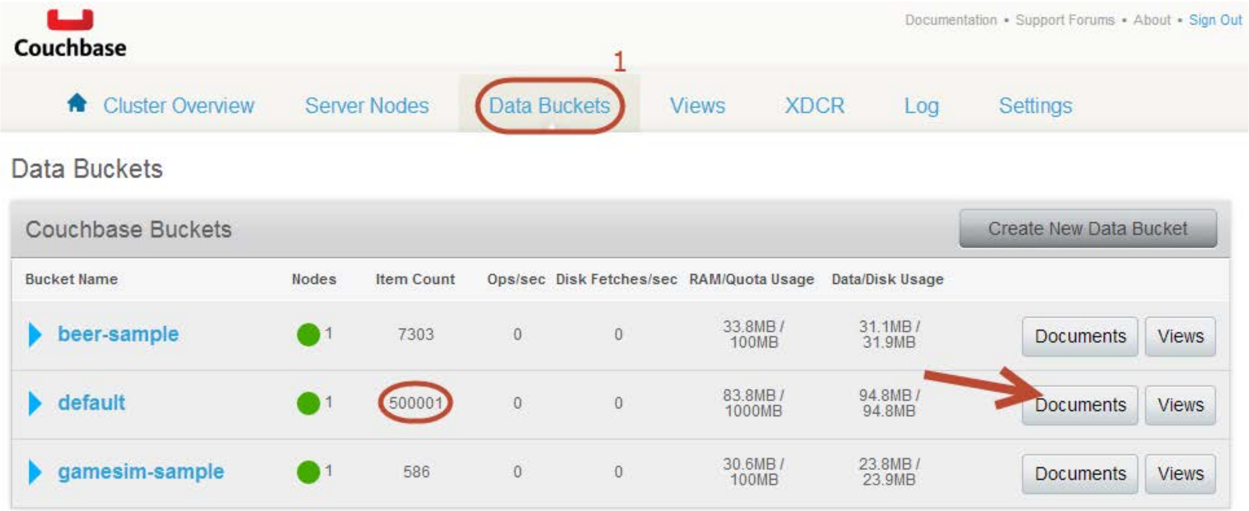
Disconnect:

```
quit
Connection closed by foreign host.
```

A fuller list with explanations of the memcached commands can be found here:
http://blog.elijaa.org/?post/2010/05/21/Memcached-telnet-command-summary

The key named test_key that we inserted via telnet above will be expiring in a few minutes, so quickly complete the next steps to verify that the key exists via the Web UI before its TTL (5 mins) runs out and it gets deleted from the memory of the Couchbase Server.

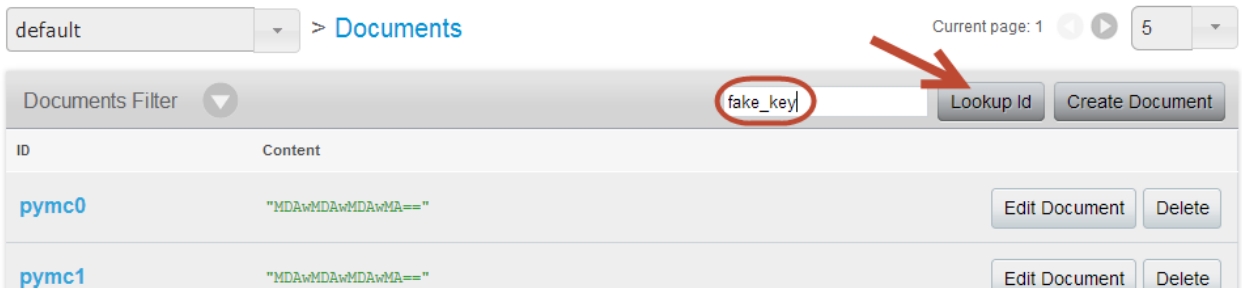
Switch to the Couchbase Web UI and click on Data Buckets at the top:



Bucket Name	Nodes	Item Count	Ops/sec	Disk Fetches/sec	RAM/Quota Usage	Data/Disk Usage		
beer-sample	● 1	7303	0	0	33.8MB / 100MB	31.1MB / 31.9MB	Documents	Views
default	● 1	500001	0	0	83.8MB / 1000MB	94.8MB / 94.8MB	Documents	Views
gamesim-sample	● 1	586	0	0	30.6MB / 100MB	23.8MB / 23.9MB	Documents	Views

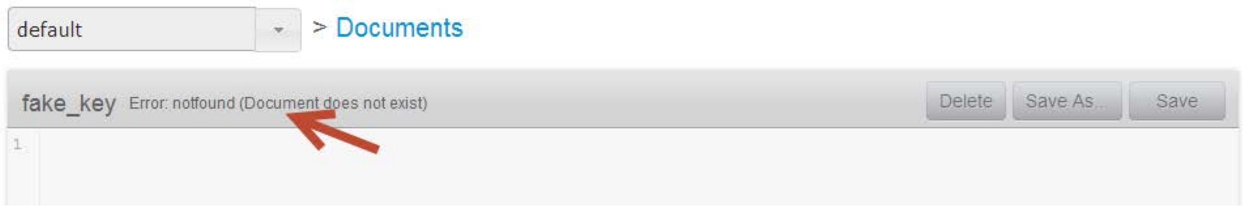
Notice that the Item Count for the default bucket is 500,001. **Click on Documents.**

Search for a fake key that doesn't exist by typing 'fake_key' into the search field and clicking 'Lookup Id':



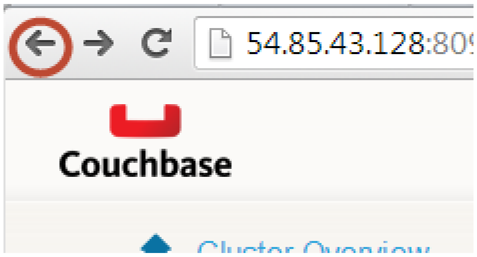
ID	Content		
pymc0	"MDAwMDAwMDAwMA=="	Edit Document	Delete
pymc1	"MDAwMDAwMDAwMA=="	Edit Document	Delete

You will see an error that the 'Document does not exist':

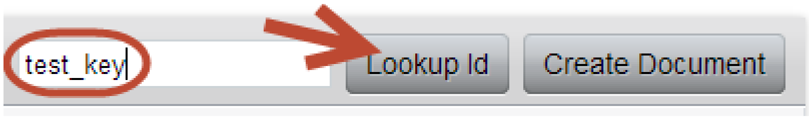


fake_key Error: notfound (Document does not exist)

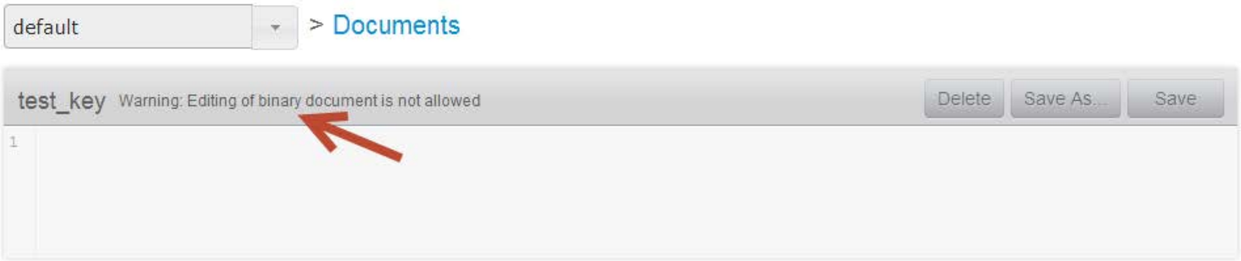
Hit back in your browser window to return to the search field:



Search for 'test_key':

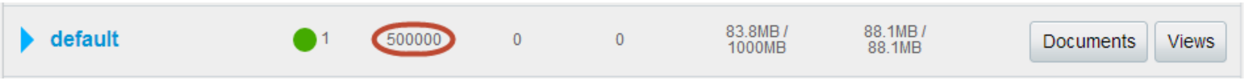


The test_key exists, however the web UI cannot display the value since it was saved as a binary document:



Warning: Editing of binary document is not allowed

Note that eventually the 5 minute Time To Live (TTL) for the test_key will expire and the document will be deleted. At this point, the Item Count for the default bucket will lower to 500,000:



default ● 1 500000 0 0 83.8MB / 1000MB 88.1MB / 88.1MB

Note : it may need a browser refresh or manual compaction to delete the key.
Click on the blue triangle next to default and hit the compact button.

Run REST API commands from App Client:

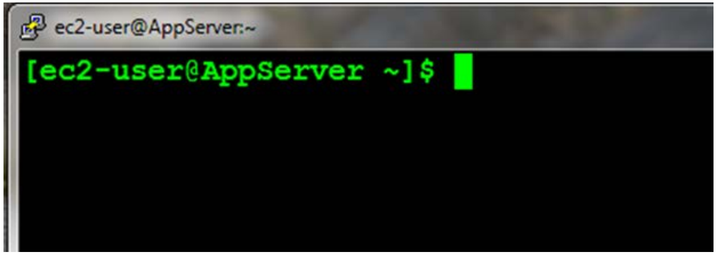
Yet another way to test Couchbase Server is to submit commands to it via the REST API.

The Couchbase REST API enables you to manage a Couchbase Server deployment as well as perform operations such as storing design documents and querying for results. Use the REST API to manage clusters, server nodes, and buckets, and to retrieve run-time statistics within your Couchbase Server deployment. As far as data I/O is concerned, it is normal to see read queries pushed via the REST API, however writes should not go through REST (use a smart client SDK instead). Smart clients automatically discover changes in the cluster using the Couchbase Management REST API.

The Couchbase Web UI uses many of the same REST API endpoints that are used for a REST API request. This is especially for administrative tasks such as creating a new bucket, adding a node to a cluster, or changing cluster settings.

Once again, remember that the REST API should *not* be used to write production data to the server. Data operations such as `set` and `get` for example, are best handled by smart client SDKs.

You should currently be logged into the App Client PuTTY/Terminal shell.



```
[ec2-user@AppServer ~]$
```

From the AppServer node, run the following command to get a JSON document back with details about the 3 buckets on the Couchbase Server (remember to change the hostname below to the public hostname of the 1-node Couchbase Server):

```
[ec2-user@AppServer ~]$ curl -u Administrator:couchbase http://ec2-54-85-43-x.compute-1.amazonaws.com:8091/pools/default/buckets/
```

```
[{"name":"beer-
sample","bucketType":"membase","authType":"sasl","saslPassword":"","proxyPort":0,"replicaIndex":f
alse,"uri":"/pools/default/buckets/beer-
sample?bucket_uuid=a54caaef3d12d39aee621a6679c79e6a","streamingUri":"/pools/default/bucketsStream
ing/beer-
sample?bucket_uuid=a54caaef3d12d39aee621a6679c79e6a","localRandomKeyUri":"/pools/default/buckets/
```

```
beer-sample/localRandomKey","controllers":{"compactAll":"/pools/default/buckets/beer-
sample/controller/compactBucket","compactDB":"/pools/default/buckets/default/controller/compactDa
tabases","purgeDeletes":"/pools/default/buckets/beer-
sample/controller/unsafePurgeBucket","startRecovery":"/pools/default/buckets/beer-
sample/controller/startRecovery"},"nodes":[{"couchApiBaseHTTPS":"https://ec2-54-85-43-
128.compute-1.amazonaws.com:18092/beer-sample","couchApiBase":"http://ec2-54-85-43-128.compute-
1.amazonaws.com:8092/beer-
sample","systemStats":{"cpu_utilization_rate":12.1212121212121212,"swap_total":0,"swap_used":0,"mem
_total":3941662720,"mem_free":2627547136},"interestingStats":{"cmd_get":0,"couch_docs_actual_disk
_size":150211758,"couch_docs_data_size":150161563,"couch_views_actual_disk_size":850902,"couch_vi
ews_data_size":782673,"curr_items":507890,"curr_items_tot":507890,"ep_bg_fetched":0,"get_hits":0,
"mem_used":155533752,"ops":0,"vb_replica_curr_items":0},"uptime":"41623","memoryTotal":3941662720
,"memoryFree":2627547136,"mcdMemoryReserved":3007,"mcdMemoryAllocated":3007,"replication":0,"clus
terMembership":"active","status":"healthy","otpNode":"ns_1@ec2-54-85-43-128.compute-
1.amazonaws.com","thisNode":true,"hostname":"ec2-54-85-43-128.compute-
1.amazonaws.com:8091","clusterCompatibility":131077,"version":"2.5.1-1083-rel-
enterprise","os":"x86_64-unknown-linux-
gnu","ports":{"httpsMgmt":18091,"httpsCAPI":18092,"sslProxy":11214,"proxy":11211,"direct":11210}}
],"stats":{"uri":"/pools/default/buckets/beer-
sample/stats","directoryURI":"/pools/default/buckets/beer-
sample/statsDirectory","nodeStatsListURI":"/pools/default/buckets/beer-
sample/nodes"},"ddocs":{"uri":"/pools/default/buckets/beer-
sample/ddocs"},"nodeLocator":"vbucket","fastWarmupSettings":false,"autoCompactionSettings":false,
"uuid":"a54caaef3d12d39aee621a6679c79e6a","vBucketServerMap":{"hashAlgorithm":"CRC","numReplicas"
:1,"serverList":["ec2-54-85-43-128.compute-1.amazonaws.com:11210"],"vBucketMap": [[0,-1],[0,-
1],[0,-1],
```

<vBucket output truncated>

```
},"replicaNumber":1,"threadsNumber":3,"quota":{"ram":104857600,"rawRAM":104857600},"basicStats":{
"quotaPercentUsed":33.82638549804688,"opsPerSec":0,"diskFetches":0,"itemCount":7303,"diskUsed":33
520307,"dataUsed":32672768,"memUsed":35469536},"bucketCapabilitiesVer":"","bucketCapabilities":["
touch","couchapi"]},{"name":"default","bucketType":"membase","authType":"sasl","saslPassword":"",
"proxyPort":0,"replicaIndex":false,"uri":"/pools/default/buckets/default?bucket_uuid=55cab122ca5c
f5825ce509b504bcf81d","streamingUri":"/pools/default/bucketsStreaming/default?bucket_uuid=55cab12
2ca5cf5825ce509b504bcf81d","localRandomKeyUri":"/pools/default/buckets/default/localRandomKey","c
ontrollers":{"flush":"/pools/default/buckets/default/controller/doFlush","compactAll":"/pools/def
ault/buckets/default/controller/compactBucket","compactDB":"/pools/default/buckets/default/contro
ller/compactDatabases","purgeDeletes":"/pools/default/buckets/default/controller/unsafePurgeBucke
t","startRecovery":"/pools/default/buckets/default/controller/startRecovery"},"nodes":[{"couchApi
BaseHTTPS":"https://ec2-54-85-43-128.compute-
1.amazonaws.com:18092/default","couchApiBase":"http://ec2-54-85-43-128.compute-
1.amazonaws.com:8092/default","systemStats":{"cpu_utilization_rate":12.1212121212121212,"swap_total
":0,"swap_used":0,"mem_total":3941662720,"mem_free":2627547136},"interestingStats":{"cmd_get":0,"
couch_docs_actual_disk_size":150211758,"couch_docs_data_size":150161563,"couch_views_actual_disk_
size":850902,"couch_views_data_size":782673,"curr_items":507890,"curr_items_tot":507890,"ep_bg_fe
tched":0,"get_hits":0,"mem_used":155533752,"ops":0,"vb_replica_curr_items":0},"uptime":"41623","m
emoryTotal":3941662720,"memoryFree":2627547136,"mcdMemoryReserved":3007,"mcdMemoryAllocated":3007
,"replication":1,"clusterMembership":"active","status":"healthy","otpNode":"ns_1@ec2-54-85-43-
128.compute-1.amazonaws.com","thisNode":true,"hostname":"ec2-54-85-43-128.compute-
1.amazonaws.com:8091","clusterCompatibility":131077,"version":"2.5.1-1083-rel-
enterprise","os":"x86_64-unknown-linux-
gnu","ports":{"httpsMgmt":18091,"httpsCAPI":18092,"sslProxy":11214,"proxy":11211,"direct":11210}}
],"stats":{"uri":"/pools/default/buckets/default/stats","directoryURI":"/pools/default/buckets/de
fault/statsDirectory","nodeStatsListURI":"/pools/default/buckets/default/nodes"},"ddocs":{"uri":"
/pools/default/buckets/default/ddocs"},"nodeLocator":"vbucket","fastWarmupSettings":false,"autoCo
mpactionSettings":false,"uuid":"55cab122ca5cf5825ce509b504bcf81d","vBucketServerMap":{"hashAlgori
thm":"CRC","numReplicas":0,"serverList":["ec2-54-85-43-128.compute-
1.amazonaws.com:11210"],"vBucketMap":
```

<vBucket output truncated>

```
},"replicaNumber":0,"threadsNumber":3,"quota":{"ram":1048576000,"rawRAM":1048576000},"basicStats"
:{"quotaPercentUsed":8.382547760009766,"opsPerSec":0,"diskFetches":0,"itemCount":500001,"diskUsed
":92468081,"dataUsed":92453888,"memUsed":87897384},"bucketCapabilitiesVer":"","bucketCapabilities
":["touch","couchapi"]},{"name":"gamesim-
sample","bucketType":"membase","authType":"sasl","saslPassword":"","proxyPort":0,"replicaIndex":f
```

```
alse,"uri":"/pools/default/buckets/gamesim-
sample?bucket_uuid=2a64e71ebb518e339c84093ff0963ade","streamingUri":"/pools/default/bucketsStream
ing/gamesim-
sample?bucket_uuid=2a64e71ebb518e339c84093ff0963ade","localRandomKeyUri":"/pools/default/buckets/
gamesim-sample/localRandomKey","controllers":{"compactAll":"/pools/default/buckets/gamesim-
sample/controller/compactBucket","compactDB":"/pools/default/buckets/default/controller/compactDa
tabases","purgeDeletes":"/pools/default/buckets/gamesim-
sample/controller/unsafePurgeBucket","startRecovery":"/pools/default/buckets/gamesim-
sample/controller/startRecovery"},"nodes":[{"couchApiBaseHTTPS":"https://ec2-54-85-43-
128.compute-1.amazonaws.com:18092/gamesim-sample","couchApiBase":"http://ec2-54-85-43-
128.compute-1.amazonaws.com:8092/gamesim-
sample","systemStats":{"cpu_utilization_rate":12.12121212121212,"swap_total":0,"swap_used":0,"mem
_total":3941662720,"mem_free":2627547136},"interestingStats":{"cmd_get":0,"couch_docs_actual_disk
_size":150211758,"couch_docs_data_size":150161563,"couch_views_actual_disk_size":850902,"couch_vi
ews_data_size":782673,"curr_items":507890,"curr_items_tot":507890,"ep_bg_fetched":0,"get_hits":0,
"mem_used":155533752,"ops":0,"vb_replica_curr_items":0},"uptime":"41623","memoryTotal":3941662720
,"memoryFree":2627547136,"mcdMemoryReserved":3007,"mcdMemoryAllocated":3007,"replication":0,"clus
terMembership":"active","status":"healthy","otpNode":"ns_1@ec2-54-85-43-128.compute-
1.amazonaws.com","thisNode":true,"hostname":"ec2-54-85-43-128.compute-
1.amazonaws.com:8091","clusterCompatibility":131077,"version":"2.5.1-1083-rel-
enterprise","os":"x86_64-unknown-linux-
gnu","ports":{"httpsMgmt":18091,"httpsCAPI":18092,"sslProxy":11214,"proxy":11211,"direct":11210}}
],"stats":{"uri":"/pools/default/buckets/gamesim-
sample/stats","directoryURI":"/pools/default/buckets/gamesim-
sample/statsDirectory","nodeStatsListURI":"/pools/default/buckets/gamesim-
sample/nodes"},"ddocs":{"uri":"/pools/default/buckets/gamesim-
sample/ddocs"},"nodeLocator":"vbucket","fastWarmupSettings":false,"autoCompactionSettings":false,
"uuid":"2a64e71ebb518e339c84093ff0963ade","vBucketServerMap":{"hashAlgorithm":"CRC","numReplicas"
:1,"serverList":["ec2-54-85-43-128.compute-1.amazonaws.com:11210"],"vBucketMap":
```

<vBucket output truncated>

```
},"replicaNumber":1,"threadsNumber":3,"quota":{"ram":104857600,"rawRAM":104857600},"basicStats":{
"quotaPercentUsed":30.67668151855469,"opsPerSec":0,"diskFetches":0,"itemCount":586,"diskUsed":250
74272,"dataUsed":25034907,"memUsed":32166832},"bucketCapabilitiesVer":"","bucketCapabilities":["t
ouch","couchapi"]}]
```

RERUN the command with output to json formatter. Like this

```
# curl -u Administrator:couchbase http://ec2-54-85-43-x.compute-
1.amazonaws.com:8091/pools/default/buckets/ | python –mjson.tool >
json_output_file

# ls
couchbase-server-enterprise-4.5.0-DP1-centos7.x86_64.rpm
json_output_file

Take a look at the file
# more json_output_file
```

The formatted JSON for the beer-sample bucket will look like this. Skim through some of the lines below to get a feel for what sort of information is returned:

```
{
   "name":"beer-sample",
   "bucketType":"membase",
   "authType":"sasl",
   "saslPassword":"",
   "proxyPort":0,
   "replicaIndex":false,
   "uri":"/pools/default/buckets/beer-sample?bucket_uuid=a54caaef3d12d39aee621a6679c79e6a",
   "streamingUri":"/pools/default/bucketsStreaming/beer-
sample?bucket_uuid=a54caaef3d12d39aee621a6679c79e6a",
   "localRandomKeyUri":"/pools/default/buckets/beer-sample/localRandomKey",
   "controllers":{
      "compactAll":"/pools/default/buckets/beer-sample/controller/compactBucket",
      "compactDB":"/pools/default/buckets/default/controller/compactDatabases",
      "purgeDeletes":"/pools/default/buckets/beer-sample/controller/unsafePurgeBucket",
      "startRecovery":"/pools/default/buckets/beer-sample/controller/startRecovery"
   },
   "nodes":[
      {
         "couchApiBaseHTTPS":"https://ec2-54-85-43-128.compute-1.amazonaws.com:18092/beer-
sample",
         "couchApiBase":"http://ec2-54-85-43-128.compute-1.amazonaws.com:8092/beer-sample",
         "systemStats":{
            "cpu_utilization_rate":12.12121212121212,
            "swap_total":0,
            "swap_used":0,
            "mem_total":3941662720,
            "mem_free":2627547136
         },
         "interestingStats":{
            "cmd_get":0,
            "couch_docs_actual_disk_size":150211758,
            "couch_docs_data_size":150161563,
            "couch_views_actual_disk_size":850902,
            "couch_views_data_size":782673,
            "curr_items":507890,
            "curr_items_tot":507890,
            "ep_bg_fetched":0,
            "get_hits":0,
            "mem_used":155533752,
            "ops":0,
            "vb_replica_curr_items":0
         },
         "uptime":"41623",
         "memoryTotal":3941662720,
         "memoryFree":2627547136,
         "mcdMemoryReserved":3007,
         "mcdMemoryAllocated":3007,
         "replication":0,
         "clusterMembership":"active",
         "status":"healthy",
         "otpNode":"ns_1@ec2-54-85-43-128.compute-1.amazonaws.com",
         "thisNode":true,
         "hostname":"ec2-54-85-43-128.compute-1.amazonaws.com:8091",
         "clusterCompatibility":131077,
         "version":"2.5.1-1083-rel-enterprise",
         "os":"x86_64-unknown-linux-gnu",
         "ports":{
            "httpsMgmt":18091,
            "httpsCAPI":18092,
            "sslProxy":11214,
            "proxy":11211,
            "direct":11210
         }
      }
   ],
   "stats":{
      "uri":"/pools/default/buckets/beer-sample/stats",
      "directoryURI":"/pools/default/buckets/beer-sample/statsDirectory",
      "nodeStatsListURI":"/pools/default/buckets/beer-sample/nodes"
   },
   "ddocs":{
      "uri":"/pools/default/buckets/beer-sample/ddocs"
   },
   "nodeLocator":"vbucket",
   "fastWarmupSettings":false,
   "autoCompactionSettings":false,
   "uuid":"a54caaef3d12d39aee621a6679c79e6a",
   "vBucketServerMap":{
      "hashAlgorithm":"CRC",
      "numReplicas":1,
      "serverList":[
         "ec2-54-85-43-128.compute-1.amazonaws.com:11210"
      ],
      "vBucketMap":[
         [
            0,
            -1
         ],
         [
            0,
            -1
         ],
         [
            0,
            -1
         ],
         <vBucket output truncated>
      },
      "replicaNumber":1,
      "threadsNumber":3,
      "quota":{
         "ram":104857600,
         "rawRAM":104857600
      },
      "basicStats":{
         "quotaPercentUsed":33.82638549804688,
         "opsPerSec":0,
         "diskFetches":0,
         "itemCount":7303,
         "diskUsed":33520307,
         "dataUsed":32672768,
         "memUsed":35469536
      },
      "bucketCapabilitiesVer":"",
      "bucketCapabilities":[
         "touch",
         "couchapi"
      ]
   }
```

You can also view cluster details by issuing the following HTTP get call:

```
[ec2-user@ AppServer ~]$ curl -u Administrator:couchbase http://ec2-54-85-43-x.compute-1.amazonaws.com:8091/pools/default| python -mjson.tool
```

Note results below have been formatted with the JSON formatter tool:

```
{
   "storageTotals":{
      "ram":{
         "total":3941662720,
         "quotaTotal":2364538880,
         "quotaUsed":1258291200,
         "used":2875891712,
         "usedByData":155533768
      },
      "hdd":{
         "total":6341722112,
         "quotaTotal":6341722112,
         "used":3044026613,
         "usedByData":151070852,
         "free":3297695499
      }
   },
   "serverGroupsUri":"/pools/default/serverGroups?v=107930833",
   "name":"default",
   "alerts":[

   ],
   "alertsSilenceURL":"/controller/resetAlerts?token=0&uuid=830b1c65e1efadd48677667bd8b8975f",
   "nodes":[
      {
         "systemStats":{
            "cpu_utilization_rate":13,
            "swap_total":0,
            "swap_used":0,
            "mem_total":3941662720,
            "mem_free":2614177792
         },
         "interestingStats":{
            "cmd_get":0,
            "couch_docs_actual_disk_size":150219950,
            "couch_docs_data_size":150169755,
            "couch_views_actual_disk_size":850902,
            "couch_views_data_size":782673,
            "curr_items":507889,
            "curr_items_tot":507889,
            "ep_bg_fetched":0,
            "get_hits":0,
            "mem_used":155533768,
            "ops":0,
            "vb_replica_curr_items":0
         },
         "uptime":"42838",
         "memoryTotal":3941662720,
         "memoryFree":2614177792,
         "mcdMemoryReserved":3007,
         "mcdMemoryAllocated":3007,
         "couchApiBase":"http://ec2-54-85-43-128.compute-1.amazonaws.com:8092/",
         "otpCookie":"cvqovpezgoidzcur",
         "clusterMembership":"active",
         "status":"healthy",
         "otpNode":"ns_1@ec2-54-85-43-128.compute-1.amazonaws.com",
         "thisNode":true,
         "hostname":"ec2-54-85-43-128.compute-1.amazonaws.com:8091",
         "clusterCompatibility":131077,
         "version":"2.5.1-1083-rel-enterprise",
         "os":"x86_64-unknown-linux-gnu",
         "ports":{
            "httpsMgmt":18091,
            "httpsCAPI":18092,
            "sslProxy":11214,
            "proxy":11211,
            "direct":11210
         }
      }
   ],
   "buckets":{
```

<output truncated>

Finally, run the following command to retrieve a list of all the nodes in this cluster (there is only 1-node at the moment):

```
[ec2-user@ AppServer ~]$ curl -u Administrator:couchbase http://ec2-54-85-43-x.compute-1.amazonaws.com:8091/pools/default/buckets/default/nodes | python -mjson.tool
{
   "servers":[
      {
         "hostname":"ec2-54-85-43-128.compute-1.amazonaws.com:8091",
         "uri":"/pools/default/buckets/default/nodes/ec2-54-85-43-128.compute-1.amazonaws.com%3A8091",
         "stats":{
            "uri":"/pools/default/buckets/default/nodes/ec2-54-85-43-128.compute-1.amazonaws.com%3A8091/stats"
         }
      }
   ]
}
```

This link contains a full reference for the Couchbase REST API:
http://docs.couchbase.com/admin/admin/rest-intro.html

older rev reference
http://docs.couchbase.com/couchbase-manual-2.5/cb-rest-api/

Install libcouchbase, run Pillow Fight and run cbc commands:

The final way we will push I/O to the Couchbase cluster is using a tool called Pillow Fight. You should currently be logged into the App Client PuTTY/Terminal shell.

```
[ec2-user@AppServer ~]$
```

First add the Couchbase repository to the CentOS package manager:

Note: If this command does not work then create the couchbase.repo and manually edit the contents.

```
 [ec2-user@AppServer ~]$ sudo wget -O
/etc/yum.repos.d/couchbase.repo http://packages.couchbase.com/rp
m/couchbase-centos72-x86_64.repo
```

Note: If this command does not work then create the couchbase.repo and manually edit the contents.

Become root

```
# sudo -i
# vi /etc/yum.repos.d/couchbase.repo
```

```
[couchbase]
enabled = 1
name = Couchbase package repository
baseurl = http://packages.couchbase.com/rpm/7/x86_64
gpgcheck = 1
gpgkey = http://packages.couchbase.com/rpm/couchbase-rpm.key
```

```
# exit
```

Then to install libcouchbase itself, update the package manager, then run the install command:

```
[ec2-user@ AppServer ~]$ sudo yum check-update
Loaded plugins: amazon-id, rhui-lb, security
couchbase                                               | 2.5 kB     00:00
couchbase/primary_db                                    | 5.2 kB     00:00
rhui-REGION-client-config-server-6                      | 2.6 kB     00:00
rhui-REGION-rhel-server-releases                        | 3.7 kB     00:00
rhui-REGION-rhel-server-releases/primary_db             |  26 MB     00:01
<output truncated>
```

```
[ec2-user@ AppServer ~]$ sudo yum install -y
libcouchbase2-libevent libcouchbase-devel libcouchbase2-bin
Loaded plugins: amazon-id, rhui-lb, search-disabled-repos
Resolving Dependencies
--> Running transaction check
---> Package libcouchbase-devel.x86_64 0:2.5.8-1.el7.centos will be installed
--> Processing Dependency: libcouchbase2-core = 2.5.8-1.el7.centos for package: libcouchbase-
devel-2.5.8-1.el7.centos.x86_64
--> Processing Dependency: libcouchbase.so.2()(64bit) for package: libcouchbase-devel-2.5.8-
1.el7.centos.x86_64
---> Package libcouchbase2-bin.x86_64 0:2.5.8-1.el7.centos will be installed
---> Package libcouchbase2-libevent.x86_64 0:2.5.8-1.el7.centos will be installed
--> Processing Dependency: libevent >= 1.4 for package: libcouchbase2-libevent-2.5.8-
1.el7.centos.x86_64
--> Processing Dependency: libevent_core-2.0.so.5()(64bit) for package: libcouchbase2-libevent-
2.5.8-1.el7.centos.x86_64
--> Running transaction check
---> Package libcouchbase2-core.x86_64 0:2.5.8-1.el7.centos will be installed
---> Package libevent.x86_64 0:2.0.21-4.el7 will be installed
--> Finished Dependency Resolution

Dependencies Resolved
```

```
=================================================================================================
Package                             Arch                Version
Repository                                     Size
=================================================================================================
Installing:
 libcouchbase-devel                  x86_64              2.5.8-1.el7.centos
couchbase                                     108 k
 libcouchbase2-bin                   x86_64              2.5.8-1.el7.centos
couchbase                                     107 k
 libcouchbase2-libevent              x86_64              2.5.8-1.el7.centos
couchbase                                     7.3 k
Installing for dependencies:
 libcouchbase2-core                  x86_64              2.5.8-1.el7.centos
couchbase                                     300 k
 libevent                            x86_64              2.0.21-4.el7                       rhui-
REGION-rhel-server-releases               214 k

Transaction Summary
=================================================================================================
Install  3 Packages (+2 Dependent packages)

Total download size: 736 k
Installed size: 2.3 M
Downloading packages:
warning: /var/cache/yum/x86_64/7Server/couchbase/packages/libcouchbase-devel-2.5.8-
1.el7.centos.x86_64.rpm: Header V4 DSA/SHA1 Signature, key ID cd406e62: NOKEY
Public key for libcouchbase-devel-2.5.8-1.el7.centos.x86_64.rpm is not installed
(1/5): libcouchbase-devel-2.5.8-1.el7.centos.x86_64.rpm   | 108 kB  00:00:00
(2/5): libcouchbase2-core-2.5.8-1.el7.centos.x86_64.rpm   | 300 kB  00:00:00
(3/5): libcouchbase2-libevent-2.5.8-1.el7.centos.x86_64.rpm  | 7.3 kB  00:00:00
(4/5): libcouchbase2-bin-2.5.8-1.el7.centos.x86_64.rpm    | 107 kB  00:00:00
(5/5): libevent-2.0.21-4.el7.x86_64.rpm                   | 214 kB  00:00:00
-------------------------------------------------------------------------------------------------
Total
1.2 MB/s | 736 kB  00:00:00
Retrieving key from http://packages.couchbase.com/rpm/couchbase-rpm.key
Importing GPG key 0xCD406E62:
 Userid     : "Couchbase Release Key (RPM) <support@couchbase.com>"
 Fingerprint: 136c d3ba 884e 3cb0 e44e 7a5b e905 c770 cd40 6e62
 From       : http://packages.couchbase.com/rpm/couchbase-rpm.key
Importing GPG key 0xD9223EDA:
 Userid     : "Couchbase Release Key <support@couchbase.com>"
 Fingerprint: 407d 39ed e720 6760 7ff1 da1c a3fa a648 d922 3eda
 From       : http://packages.couchbase.com/rpm/couchbase-rpm.key
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : libcouchbase2-core-2.5.8-1.el7.centos.x86_64                                 1/5
  Installing : libevent-2.0.21-4.el7.x86_64                                                 2/5
  Installing : libcouchbase2-libevent-2.5.8-1.el7.centos.x86_64                             3/5
  Installing : libcouchbase2-bin-2.5.8-1.el7.centos.x86_64                                  4/5
  Installing : libcouchbase-devel-2.5.8-1.el7.centos.x86_64                                 5/5
  Verifying  : libcouchbase2-core-2.5.8-1.el7.centos.x86_64                                 1/5
  Verifying  : libcouchbase2-libevent-2.5.8-1.el7.centos.x86_64                             2/5
  Verifying  : libcouchbase2-bin-2.5.8-1.el7.centos.x86_64                                  3/5
  Verifying  : libevent-2.0.21-4.el7.x86_64                                                 4/5
  Verifying  : libcouchbase-devel-2.5.8-1.el7.centos.x86_64                                 5/5

Installed:
  libcouchbase-devel.x86_64 0:2.5.8-1.el7.centos  libcouchbase2-bin.x86_64 0:2.5.8-1.el7.centos
  libcouchbase2-libevent.x86_64 0:2.5.8-1.el7.centos

Dependency Installed:
  libcouchbase2-core.x86_64 0:2.5.8-1.el7.centos         libevent.x86_64 0:2.0.21-4.el7

Complete!
```

The syntax for running Pillow Fight is as follows:

```
cbc pillowfight [-?] [-h HOST] [-b BUCKET] [-u USER] [-P PASSWORD] [-
T] [-i ITERATIONS] [-I ITEMS] [-p PREFIX] [-t THREADS] [-Q INSTANCES]
[-l] [-s SEED] [-r RATIO] [-m MIN] [-M MAX] [-d]
```

Notice that pillowfight is a subcommand of cbc, the Couchbase Command Line Utility. Print the help menu for pillow fight:

```
[ec2-user@ AppServer ~]$ cbc-pillowfight -?
Usage:cbc-pillowfight [OPTIONS...]

-B  --batch-size   Number of operations to batch [Default=100]
-I  --num-items    Number of items to operate on [Default=1000]
-p  --key-prefix   key prefix to use [Default='']
-t  --num-threads  The number of threads to use [Default=1]
-r  --set-pct     The percentage of operations which should be mutations [Default=33]
-n  --no-population  Skip population [Default=FALSE]
-m  --min-size    Set minimum payload size [Default=50]
-M  --max-size      Set maximum payload size [Default=5120]
-E  --pause-at-end  Pause at end of run (holding connections open) until user input
[Default=FALSE]
-c  --num-cycles    Number of cycles to be run until exiting. Set to -1 to loop
infinitely [Default=1]
-P  --password      Bucket password [Default='']
-u  --username      Username (currently unused) [Default='']
-Z  --config-cache  Path to cached configuration [Default='']
-U  --spec    [Default='']
    --ssl  <ON|OFF|NOVERIFY>  Enable SSL settings [Default='off']
    --certpath              Path to server certificate [Default='']
-T  --timings               Enable command timings [Default=FALSE]
-v  --verbose   Set debugging output (specify multiple times for greater verbosity
[Default=FALSE]
   --dump Dump verbose internal state after operations are done [Default=FALSE]
-D  --cparam  <OPTION=VALUE>      Additional options for connection [Default=]
-?  --help          this message
```

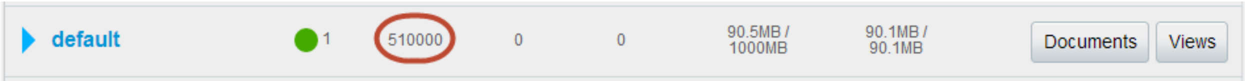
Run pillowfight to operate on 10,000 items, in 1000 iterations, with a 50% set/get ratio and a maximum payload size of 400 bytes and enable timings histograms. Use the public hostname of the 1st node in the command:

```
[ec2-user@ AppServer  ~]$  cbc-pillowfight -U couchbase://ec2-54-172-
130-66.compute-1.amazonaws.com/default  --num-items=10000 --batch-
size=20 --set-pct=50 --max-size=400 --num-cycles=1000 --timings

Creating instance 0
[1413496946.476260] Running. Press Ctrl-C to terminate...
[1413496952.952200] Populate
              +---------+---------+---------+---------+
[310  - 319 ]us |## - 67
[320  - 329 ]us |################ - 523
[330  - 339 ]us |################################## - 1091
[340  - 349 ]us |###################################### - 1195
[350  - 359 ]us |######################################## - 1247
[360  - 369 ]us |###################################### - 1195
```

```
[370  - 379 ]us |################################# - 1046
[380  - 389 ]us |######################## - 769
[390  - 399 ]us |################ - 518
[400  - 409 ]us |########## - 333
[410  - 419 ]us |###### - 204
[420  - 429 ]us |##### - 160
[430  - 439 ]us |### - 120
[440  - 449 ]us |### - 95
[450  - 459 ]us |## - 83
[460  - 469 ]us |## - 67
[470  - 479 ]us |# - 52
[480  - 489 ]us |# - 45
[490  - 499 ]us |# - 43
[500  - 509 ]us |# - 39
[510  - 519 ]us |# - 40
[520  - 529 ]us |# - 38
[530  - 539 ]us |# - 34
[540  - 549 ]us |# - 33
[550  - 559 ]us | - 28
[560  - 569 ]us |# - 36
[570  - 579 ]us | - 26
[580  - 589 ]us | - 28
[590  - 599 ]us | - 22
[600  - 609 ]us | - 21
[610  - 619 ]us | - 14
[620  - 629 ]us | - 19
[630  - 639 ]us | - 13
[640  - 649 ]us | - 19
[650  - 659 ]us | - 12
[660  - 669 ]us | - 13
[670  - 679 ]us | - 10
<output truncated>
[10000 - 10099]us | - 2
[10200 - 10299]us | - 4
[10400 - 10499]us | - 2
[10500 - 10599]us | - 2
[10600 - 10699]us | - 2
[10700 - 10799]us | - 1
[10   - 19  ]ms |# - 38
[20   - 29  ]ms | - 6
[30   - 39  ]ms | - 3
```

In the output histograms, you can see the time that most of the operations complete
Note : This command will load 1000 items and then iterate individually on each of these items.It will issue 50% get operations and 50% set operations on the cluster. **You should see 10,000 new items in the default bucket.**



default	● 1	510000	0	0	90.5MB / 1000MB	90.1MB / 90.1MB	Documents	Views

More information about the cbc subcommands including pillowfight can be found here:
http://www.couchbase.com/autodocs/couchbase-c-client-2.1.1/cbc.1.html

http://blog.couchbase.com/couchbase-tools-shipped-couchbase-c-client-library-libcouchbase

You can also use the cbc command to insert a key into Couchbase (run the command from the App Server, but run it against the public hostname of Node #1):

```
[ec2-user@ AppServer ~]$ cbc-create  -U couchbase://ec2-54-172-130-
66.compute-1.amazonaws.com/default -f 555 cbc_key <hit enter>
newdata<hit CTRL + D one time on the keyboard to send an EOF
character>
cbc_key             Stored. CAS=0x469e6b1544360200
```

Retrieve the cbc_key:

```
[ec2-user@ AppServer ~]$ cbc-cat -U couchbase://ec2-54-172-130-
66.compute-1.amazonaws.com/default cbc_key
cbc_key              CAS=0x469e6b1544360200, Flags=0x22b, Datatype=0x0
```

Delete the cbc_key:

```
[ec2-user@ AppServer ~]$ cbc-rm -U couchbase://ec2-54-172-130-
66.compute-1.amazonaws.com/default cbc_key
cbc_key             Deleted. CAS=0x479e6b1544360200
```

In Summary, the AppServer has established connectivity to the 1-node Couchbase Server via cbworkloadgen, telnet, REST API and cbc pillowfight.

This concludes lab #2.