CS300 Couchbase NoSQL Server Administration

Lab 2 Exercise Manual

Couchbase Learning Services

Release: 6.5.1

Revised: June 22nd, 2020

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
- 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
```

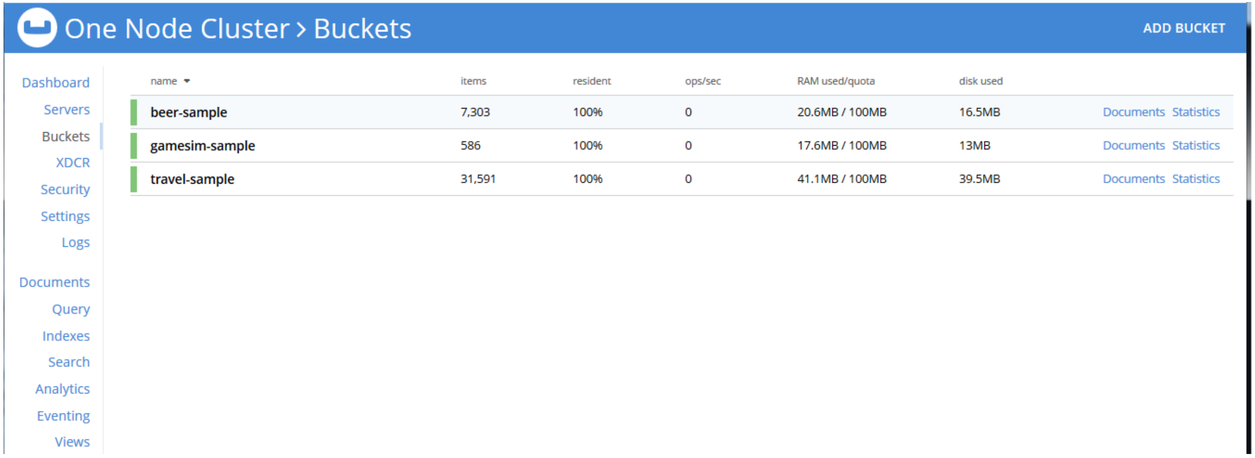
Let’s test the installation of Couchbase Server by using cbworkloadgen to insert some random data into the cluster.

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:

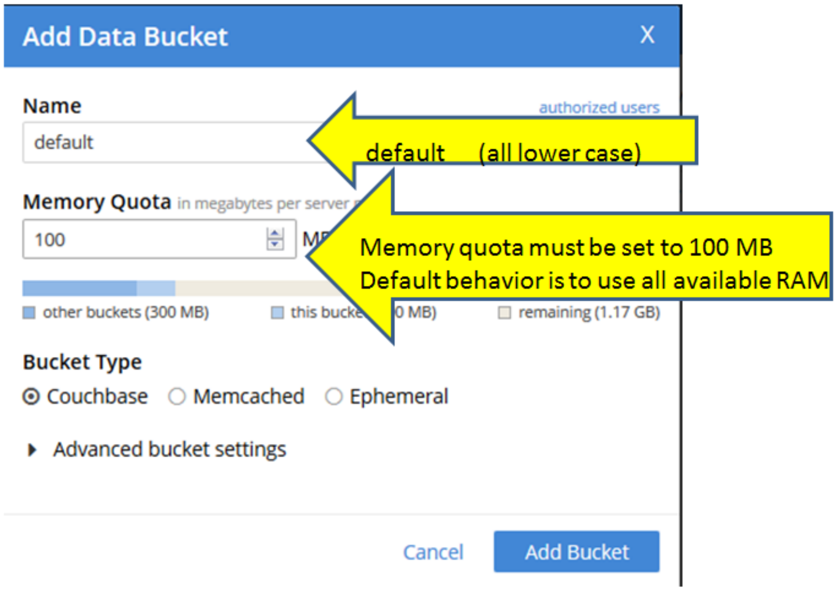
```
[ec2-user@Couchbase01 ~]$ cbworkloadgen --help
Usage: cbworkloadgen [options]

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
  --xattr               generate extended attributes for inserted documents
```

On the Buckets link page select the ADD BUCKET LINK



name	items	resident	ops/sec	RAM used/quota	disk used	
beer-sample	7,303	100%	0	20.6MB / 100MB	16.5MB	Documents Statistics
gamesim-sample	586	100%	0	17.6MB / 100MB	13MB	Documents Statistics
travel-sample	31,591	100%	0	41.1MB / 100MB	39.5MB	Documents Statistics

In the pop up



Name: default

Memory Quota in megabytes per server: 100 MB

Bucket Type: ● Couchbase ○ Memcached ○ Ephemeral

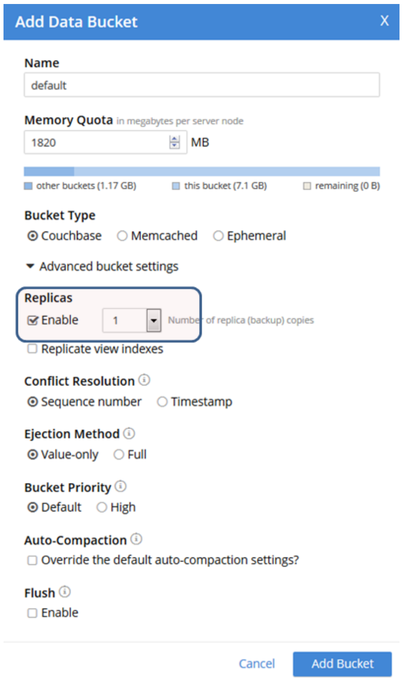
default (all lower case)

Memory quota must be set to 100 MB
Default behavior is to use all available RAM

Add the following(must be lower case):

Name	**default**
Memory Quota	**100 MB**

Click Advanced bucket settings



Add Data Bucket X

Name
default

Memory Quota in megabytes per server node
1820 MB

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

Bucket Type
● Couchbase ○ Memcached ○ Ephemeral

▾ Advanced bucket settings

Replicas
☑ Enable 1 Number of replica (backup) copies
□ Replicate view indexes

Conflict Resolution ⓘ
● Sequence number ○ Timestamp

Ejection Method ⓘ
● Value-only ○ Full

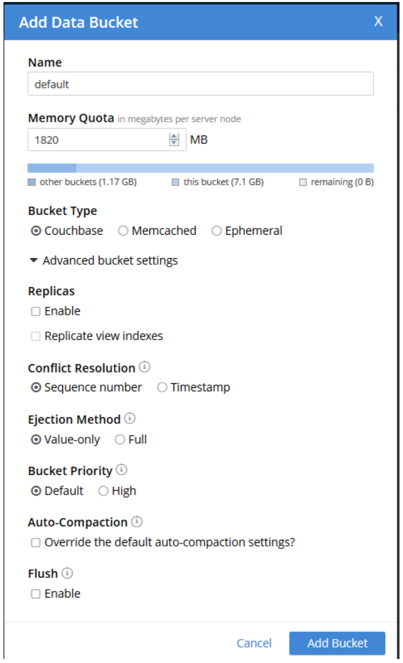
Bucket Priority ⓘ
● Default ○ High

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

Flush ⓘ
□ Enable

Cancel Add Bucket

Deselect Replicas enabled box



Add Data Bucket X

Name
default

Memory Quota in megabytes per server node
1820 MB

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

Bucket Type
● Couchbase ○ Memcached ○ Ephemeral

▾ Advanced bucket settings

Replicas
□ Enable
□ Replicate view indexes

Conflict Resolution ⓘ
● Sequence number ○ Timestamp

Ejection Method ⓘ
● Value-only ○ Full

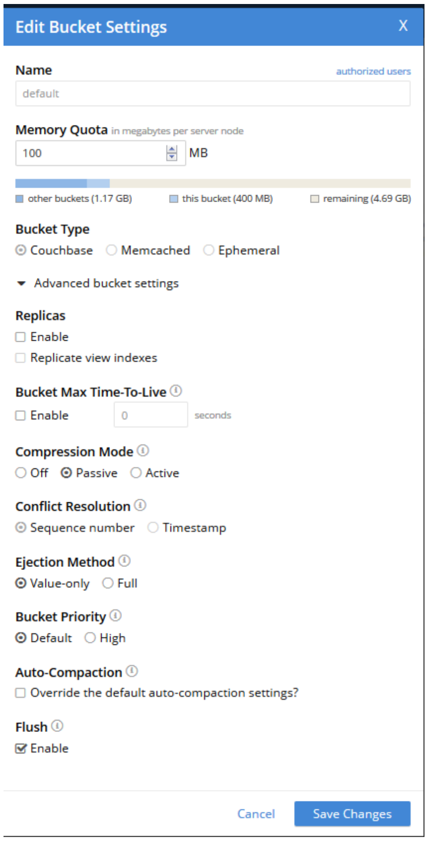
Bucket Priority ⓘ
● Default ○ High

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

Flush ⓘ
□ Enable

Cancel Add Bucket

Lastly, select flush enable box



Edit Bucket Settings X

Name authorized users

default

Memory Quota in megabytes per server node

100 MB

▨ other buckets (1.17 GB) ▨ this bucket (400 MB) □ remaining (4.69 GB)

Bucket Type

● Couchbase ○ Memcached ○ Ephemeral

▾ Advanced bucket settings

Replicas

□ Enable

□ Replicate view indexes

Bucket Max Time-To-Live ⓘ

□ Enable 0 seconds

Compression Mode ⓘ

○ Off ● Passive ○ Active

Conflict Resolution ⓘ

● Sequence number ○ Timestamp

Ejection Method ⓘ

● Value-only ○ Full

Bucket Priority ⓘ

● Default ○ High

Auto-Compaction ⓘ

□ Override the default auto-compaction settings?

Flush ⓘ

☑ Enable

Click

Run cbworkloadgen with localhost, username & password:

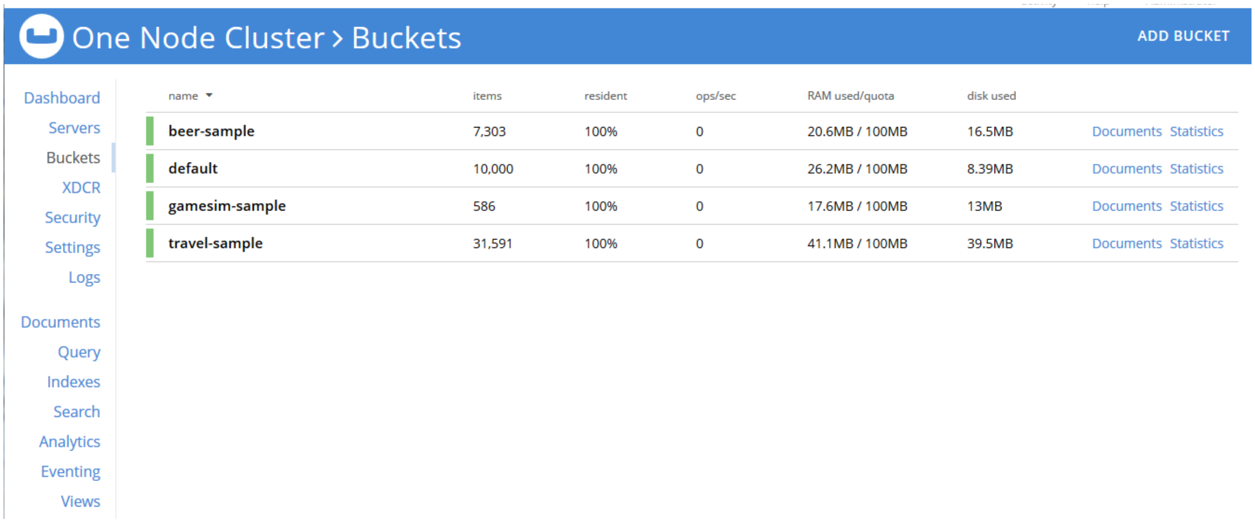
```
[ec2-user@Couchbase01 ~]$ cbworkloadgen -n $NODE1:8091
[####################] 100.0% (10527/estimated 10526 msgs)
bucket: default, msgs transferred...
       :                total |       last |    per sec
 byte  :               105270 |     105270 |    217014.8
done
```

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

NOTE: **username & password not required at command line since set as environmental parameter.** `-u Administrator -p couchbase`

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



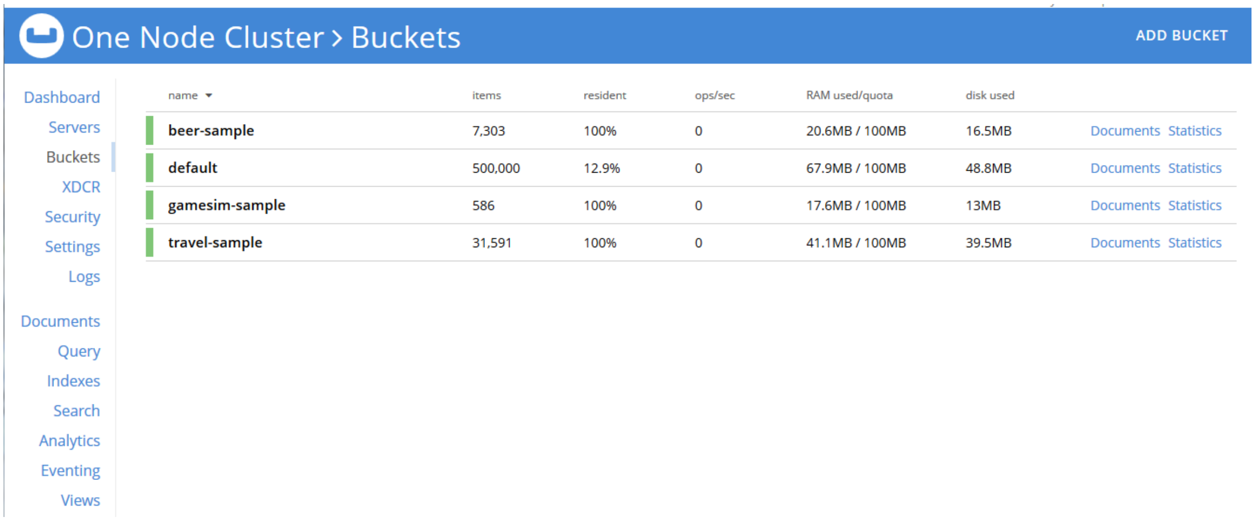
name	items	resident	ops/sec	RAM used/quota	disk used	
beer-sample	7,303	100%	0	20.6MB / 100MB	16.5MB	Documents Statistics
default	10,000	100%	0	26.2MB / 100MB	8.39MB	Documents Statistics
gamesim-sample	586	100%	0	17.6MB / 100MB	13MB	Documents Statistics
travel-sample	31,591	100%	0	41.1MB / 100MB	39.5MB	Documents Statistics

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 $NODE1:8091 -i 500000 -r .5
-s 10
2017-09-26 19:41:06,474: s0 backing off, secs: 0.1000000 msgs)
2017-09-26 19:41:07,416: s0 backing off, secs: 0.1000000 msgs)
  [####################] 100.0% (999999/estimated 1000000 msgs)
bucket: default, msgs transferred...
             :               total |       last |    per sec
 byte        :             9999990 |    9999990 |   332161.2
 retry_batch :                   2 |          2 |        0.1
done
```

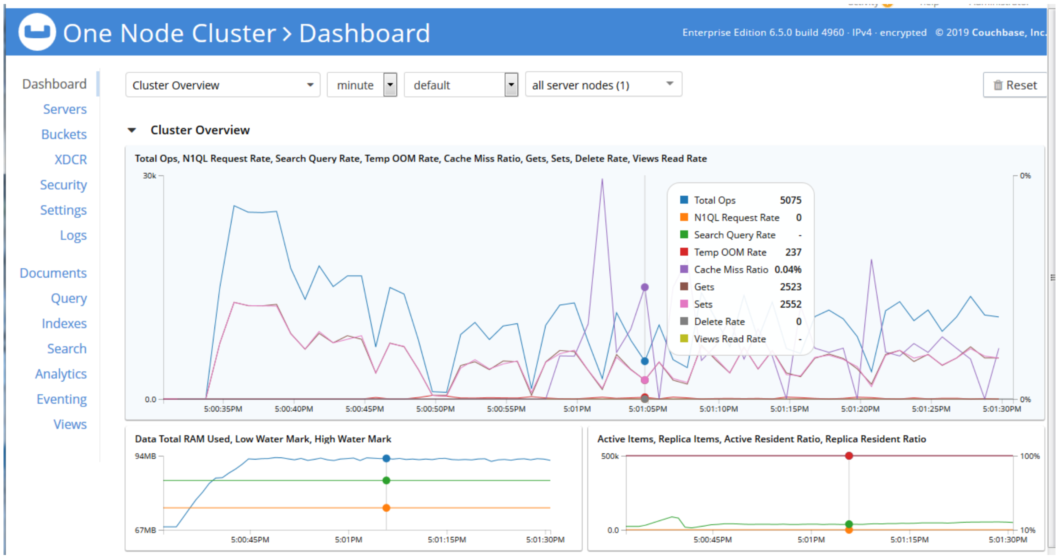
The above command will take about 30-35 seconds to run. While this workload is running, quickly continue with the next few steps. First refresh the Couchbase Web UI and you should see more items added to the default bucket



One Node Cluster > Buckets

name	items	resident	ops/sec	RAM used/quota	disk used		
beer-sample	7,303	100%	0	20.6MB / 100MB	16.5MB	Documents	Statistics
default	500,000	12.9%	0	67.9MB / 100MB	48.8MB	Documents	Statistics
gamesim-sample	586	100%	0	17.6MB / 100MB	13MB	Documents	Statistics
travel-sample	31,591	100%	0	41.1MB / 100MB	39.5MB	Documents	Statistics

Click on 'Dashboard" link on the left side of the screen, cluster overview selected form pulldown menu , minute for period, default bucket selected across all server nodes:



One Node Cluster > Dashboard

Enterprise Edition 6.5.0 build 4960 · IPv4 · encrypted © 2019 Couchbase, Inc.

Cluster Overview | minute | default | all server nodes (1)

Cluster Overview

Total Ops, N1QL Request Rate, Search Query Rate, Temp OOM Rate, Cache Miss Ratio, Gets, Sets, Delete Rate, Views Read Rate

Total Ops	5075
N1QL Request Rate	0
Search Query Rate	-
Temp OOM Rate	237
Cache Miss Ratio	0.04%
Gets	2523
Sets	2552
Delete Rate	0
Views Read Rate	-

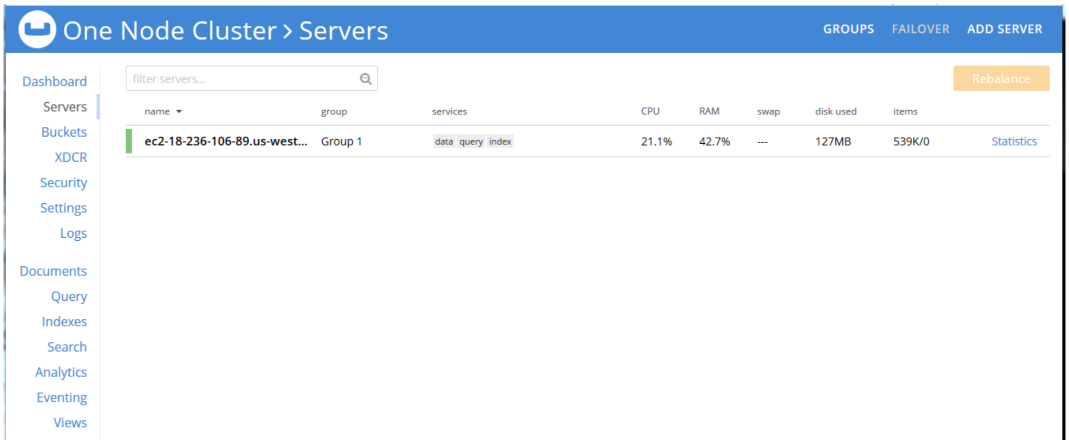
Data Total RAM Used, Low Water Mark, High Water Mark

Active Items, Replica Items, Active Resident Ratio, Replica Resident Ratio

Observe the Ops per second graph:

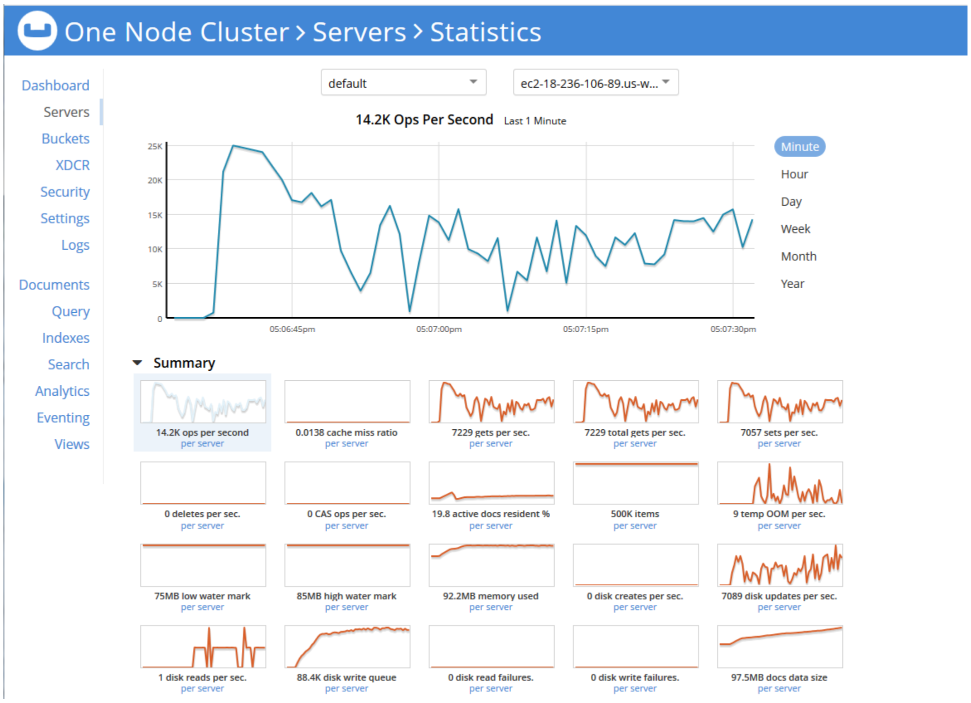
Notice operations per second. In your specific cloud environment, the range of ops per second may vary.

You can get more detailed performance graphs, by clicking on "Server" link at the left hand side and then clicking on the specific server's statistics link   on the right hand side of the page:



name	group	services	CPU	RAM	swap	disk used	items	
ec2-18-236-106-89.us-west...	Group 1	data query index	21.1%	42.7%	---	127MB	539K/0	Statistics

Select the default bucket and the resulting page will look like this:


One Node Cluster > Servers > Statistics
Dashboard
Servers
Buckets
XDCR
Security
Settings
Logs
Documents
Query
Indexes
Search
Analytics
Eventing
Views
default
ec2-18-236-106-89.us-w...
14.2K Ops Per Second Last 1 Minute
25K
20K
15K
10K
5K
0
05:06:45pm
05:07:00pm
05:07:15pm
05:07:30pm
Minute
Hour
Day
Week
Month
Year
Summary
14.2K ops per second per server
0.0138 cache miss ratio per server
7229 gets per sec. per server
7229 total gets per sec. per server
7057 sets per sec. per server
0 deletes per sec. per server
0 CAS ops per sec. per server
19.8 active docs resident % per server
500K items per server
9 temp OOM per sec. per server
75MB low water mark per server
85MB high water mark per server
92.2MB memory used per server
0 disk creates per sec. per server
7089 disk updates per sec. per server
1 disk reads per sec. per server
88.4K disk write queue per server
0 disk read failures. per server
0 disk write failures. per server
97.5MB docs data size per server

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:

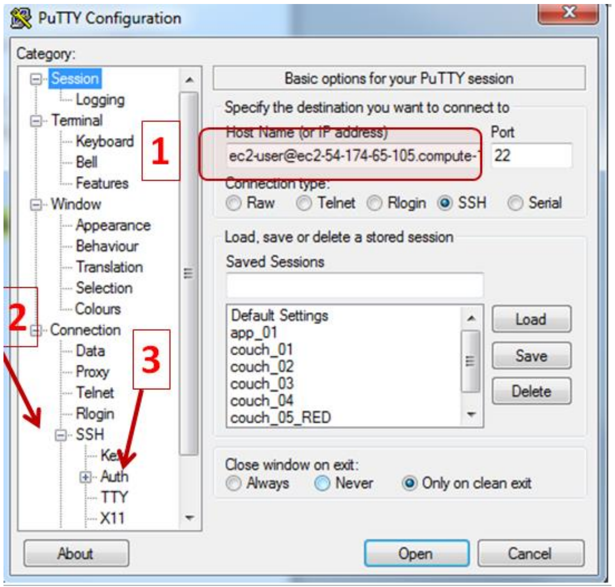
Now that we have verified that Couchbase Server is working and accepting fresh writes from a local client, next we will set up and configure a new client application server.

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: **20 GB**
Network performance: **moderate**
CloudWatch Monitoring: **disabled**
Tenancy: **Shared tenancy** (multi-tenant hardware)
Cost: **$0.11 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-
Port: 22

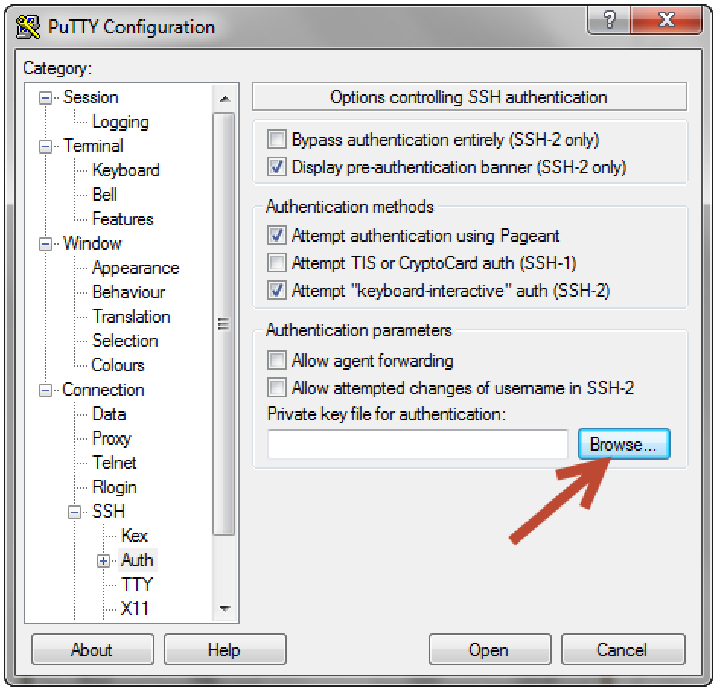
Connection type:
○ Raw ○ Telnet ○ Rlogin ● SSH ○ Serial

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

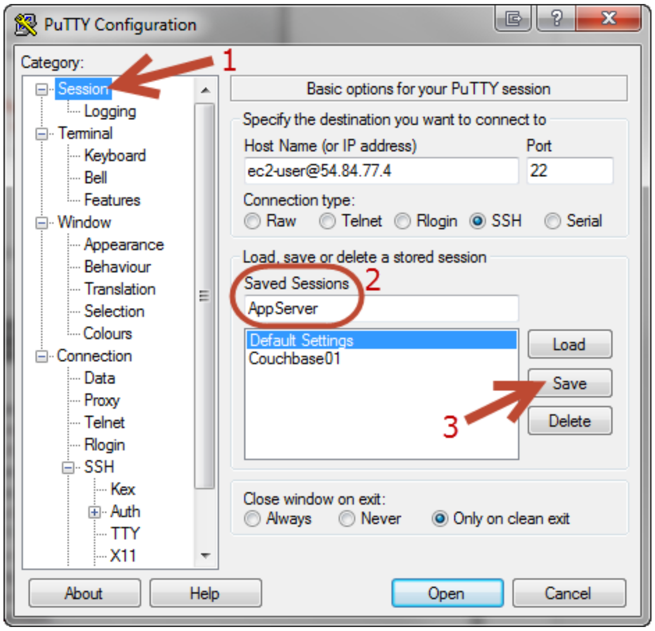
☑ Attempt authentication using Pageant
□ Attempt TIS or CryptoCard auth (SSH-1)
☑ Attempt "keyboard-interactive" auth (SSH-2)

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

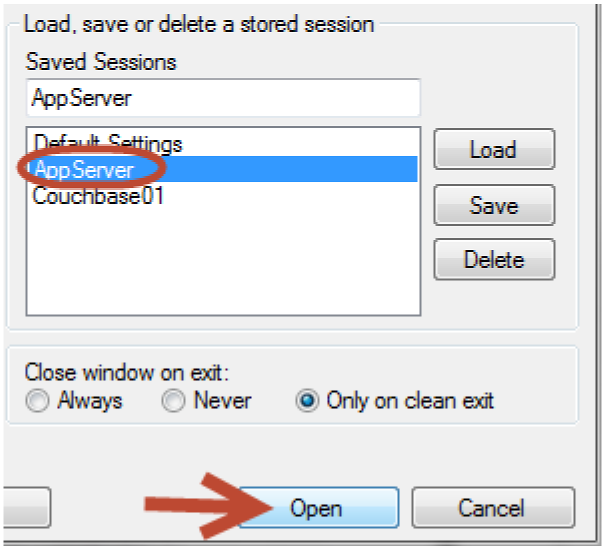
Load, save or delete a stored session

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

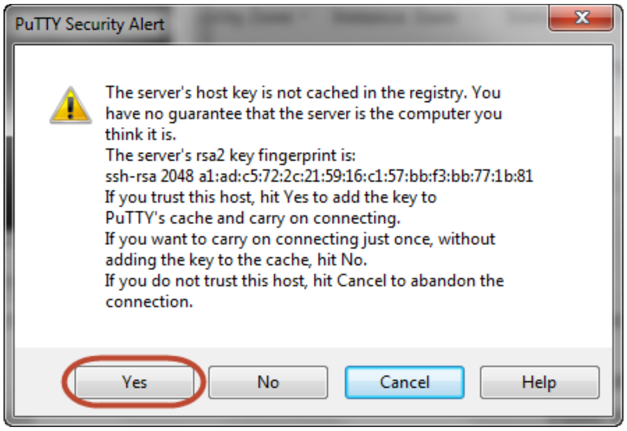
Saved Sessions

AppServer

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 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

```
# hostnamectl set-hostname AppServer
# hostnamectl status
```

```
Static hostname: AppServer
         Icon name: computer-vm
           Chassis: vm
        Machine ID: 4e7e0a894c2447ff9b480516eae4d8e7
           Boot ID: 4f0c256433074176beeb2b55785938e3
    Virtualization: xen
  Operating System: Red Hat Enterprise Linux 8.0 (Ootpa)
       CPE OS Name: cpe:/o:redhat:enterprise_linux:8.0:GA
            Kernel: Linux 4.18.0-80.4.2.el8_0.x86_64
      Architecture: x86-64
```

Install wget, bzip2 & python3

```
# yum install wget

# yum install bzip2

# yum install python3

# exit
```

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

Download Couchbase 6.5.1 EE (*do not copy + paste this command!*):

```
[ec2-user@AppServer ~]$  wget
http://packages.couchbase.com/releases/6.5.1/couchbase-server-
enterprise-6.5.1-centos8.x86_64.rpm
--2020-01-22 01:16:42--  http://packages.couchbase.com/releases/6.5.1/couchbase-          server-
enterprise-6.5.1-centos8.x86_64.rpm
Resolving packages.couchbase.com (packages.couchbase.com)... 13.224.2.9, 13.224.          2.30,
13.224.2.111, ...
Connecting to packages.couchbase.com (packages.couchbase.com)|13.224.2.9|:80...
connected.
HTTP request sent, awaiting response... 200 OK
Length: 394755748 (376M) [application/x-rpm]
Saving to: âcouchbase-server-enterprise-6.5.1-centos8.x86_64.rpmâ

couchbase-server-en 100%[===================>] 376.47M  38.8MB/s    in 11s

2020-01-22 01:16:53 (34.2 MB/s) - âcouchbase-server-enterprise-6.5.1-centos8.x86
_64.rpmâ saved [394755748/394755748]
```

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

```
[ec2-user@ AppServer ~]$ sudo rpm --install couchbase-server-
enterprise-6.5.1-centos8.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 : 3.70 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 systemctl status couchbase-server
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 systemctl stop couchbase-server

[ec2-user@ AppServer ~]$ sudo systemctl status couchbase-server
Jan 10 17:14:18 AppServer systemd[1]: Stopped Couchbase Server.
Hint: Some lines were ellipsized, use -l to show in full.
```

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 ~]$
```

Edit the .bashrc file:

```
[ec2-user@ Couchbase01 ~]$ cd ~
[ec2-user@ Couchbase01 ~]$ vi .bashrc
```

Line 10 should currently show the following:

```
PATH=$PATH:$HOME/.local/bin:$HOME/bin
```

Edit line 9 by appending the couchbase tools path to the end of the line, like so:

```
PATH=$PATH:$HOME/.local/bin:$HOME/bin:/opt/couchbase/bin
```

USING AMAZON worldwide DNS names provided at the start of this course by your instructor on the .XLS spreadsheet

Make make the following edits(add lines)…:

```
NODE1=ec2-113-156-188-191.us-west-1.compute.amazonaws.com
NODE2=ec2-113-156-209-124.us-west-1.compute.amazonaws.com
NODE3=ec2-113-157-200-184.us-west-1.compute.amazonaws.com
NODE4=ec2-118-144-149-199.us-west-1.compute.amazonaws.com
NODE5=ec2-154-183-155-127.us-west-1.compute.amazonaws.com
NODE6=ec2-154-183-183-230.us-west-1.compute.amazonaws.com
NODE7=ec2-154-219-170-126.us-west-1.compute.amazonaws.com
NODE8=ec2-154-219-174-110.us-west-1.compute.amazonaws.com

export PATH NODE1 NODE2 NODE3 NODE4 NODE5 NODE6 NODE7 NODE8
CB_REST_USERNAME=Administrator
CB_REST_PASSWORD=couchbase
export CB_REST_USERNAME CB_REST_PASSWORD
```

USING AMAZON worldwide DNS names provided at the start of this course

Save and quit the vi or nano session.

Source the .bash_profile file so that the changes you made take effect in the current bash session:

```
[ec2-user@ Couchbase01 ~]$ source ~/.bashrc
```

```
[ec2-user@ AppServer ]$ cbworkloadgen -n $NODE1:8091
  [####################] 100.0% (10527/estimated 10526 msgs)
bucket: default, msgs transferred...
       :                total |       last |    per sec
 byte  :               105270 |     105270 |   290846.9

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 ]$ cbworkloadgen -n $NODE1:8091 -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 REST API commands from App Client:

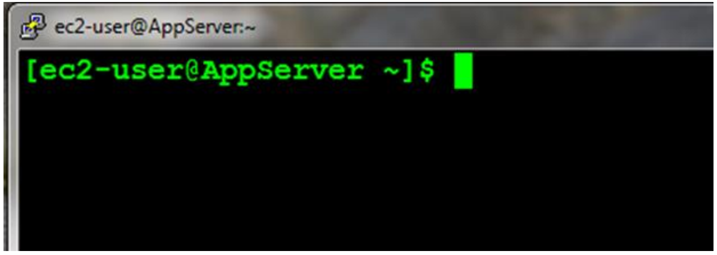
Yet another way to test Couchbase Server is to submit commands to it via the REST API.

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 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 -n
http://$NODE1:8091/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":12.12121212121212,"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/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":12.12121212121212,"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_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

```
[ec2-user@AppServer ~]$  curl -u Administrator:couchbase -n
http://$NODE1:8091/pools/default/buckets/ | python3 -mjson.tool >
json_output_file

[ec2-user@AppServer ~]$  ls
couchbase-server-enterprise-6.5.1-centos7.x86_64.rpm
json_output_file

Take a look at the file
[ec2-user@AppServer ~]$  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 -n
http://$NODE1:8091/pools/default | python3 -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 -n
http://$NODE1:8091/pools/nodes | python3 -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:

```
Become root

[root@AppServer ~]# sudo -i
[root@AppServer ~]# vi /etc/yum.repos.d/couchbase.repo
```

```
[couchbase]
enabled = 1
name = libcouchbase package for centos8 x86_64
baseurl = https://packages.couchbase.com/clients/c/repos/rpm/el8/x86_64
gpgcheck = 1
gpgkey = https://packages.couchbase.com/clients/c/repos/rpm/couchbase.key
```

# exit

```
[ec2-user@ AppServer ~]$ sudo yum install -y libcouchbase3-tools-3.0.0-
1.el8.x86_64  libcouchbase3-libevent-3.0.0-1.el8.x86_64
libcouchbase3-3.0.0-1.el8.x86_64 libcouchbase-devel-3.0.0-1.el8.x86_64
```

```
Last metadata expiration check: 0:05:50 ago on Wed 22 Jan 2020 03:27:17 AM UTC.
Dependencies resolved.
=================================================================================================
==============================
 Package                              Arch                   Version
Repository                  Size
=================================================================================================
==============================
Installing:
 libcouchbase3-tools                  x86_64                 3.0.0-1.el8
couchbase                  373 k
Installing dependencies:
 libcouchbase3                        x86_64                 3.0.0-1.el8
couchbase                  522 k

Transaction Summary
=================================================================================================
==============================
Install  2 Packages

Total download size: 895 k
Installed size: 3.5 M
Downloading Packages:
(1/2): libcouchbase3-tools-3.0.0-1.el8.x86_64.rpm
211 kB/s | 373 kB     00:01
(2/2): libcouchbase3-3.0.0-1.el8.x86_64.rpm
295 kB/s | 522 kB     00:01
-------------------------------------------------------------------------------------------------
------------------------------
Total
505 kB/s | 895 kB     00:01
warning: /var/cache/dnf/couchbase-94374a010744b981/packages/libcouchbase3-3.0.0-1.el8.x86_64.rpm:
Header V4 RSA/SHA1 Signature, key ID c4a088b2: NOKEY
libcouchbase package for centos8 x86_64
7.1 kB/s | 3.1 kB     00:00
Importing GPG key 0xC4A088B2:
 Userid     : "Couchbase SDK Team (Created for SDK Git Signing) <sdk_dev@couchbase.com>"
 Fingerprint: 5098 4187 E4FC D540 EF74 6178 1616 981C C4A0 88B2
 From       : https://packages.couchbase.com/clients/c/repos/rpm/couchbase.key
Key imported successfully
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :                                                                 1/1
  Installing       : libcouchbase3-3.0.0-1.el8.x86_64               1/2
  Running scriptlet: libcouchbase3-3.0.0-1.el8.x86_64                                1/2
  Installing       : libcouchbase3-tools-3.0.0-1.el8.x86_64                             2/2
  Running scriptlet: libcouchbase3-tools-3.0.0-1.el8.x86_64                                2/2
  Verifying        : libcouchbase3-3.0.0-1.el8.x86_64                                 1/2
  Verifying        : libcouchbase3-tools-3.0.0-1.el8.x86_64                                   2/2

Installed:
  libcouchbase3-tools-3.0.0-1.el8.x86_64           libcouchbase3-3.0.0-1.el8.x86_64



Complete!
```

Now run the cbc command

```
[ec2-user@appserver ~]$ cbc
Must provide an option name
Usage: cbc <command> [options]
command may be:
   help                Show help
   cat                 Retrieve items from the cluster
   create              Store item to the server
   touch               Updated expiry times for documents
   observe          Obtain persistence and replication status for keys
   observe-seqno   Request information about a particular vBucket UUID
   incr                Increment a counter
   decr                Decrement a counter
   mcflush             Flush a memcached bucket
   hash                Get mapping information for keys
   lock                Lock keys and retrieve them from the cluster
   unlock              Unlock keys
   cp                  Store files to the server
   rm                  Remove items from the cluster
   stats               Retrieve cluster statistics
   version             Display information about libcouchbase
   verbosity           Modify the memcached logging level
   view                Query a view
   query               Execute a N1QL/Analytics Query
   admin               Invoke an administrative REST API
   bucket-create       Create a bucket
   bucket-delete       Delete a bucket
   bucket-flush        Flush a bucket
   role-list           List roles
   user-list           List users
   user-upsert         Create or update a user
   user-delete         Delete a user
   connstr    Parse a cnction strng and provide info on its components
   write-config        Write the config file based on arguments passed
   strerror            Decode library error code
   ping                Rch all svces on evrynode & measre respnse time
   watch               Aggregate and display server statistics
   keygen              Otpt a list o/keys that equally distrib amngst
every vbucket
```

Now run the pillowfight command

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]
```

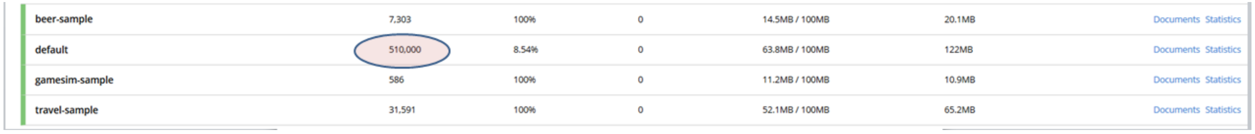
Notice that pillowfight is a subcommand of cbc, the Couchbase Command Line Utility. Print the help menu for pillow fight:

```
[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  --random-body     Randomize document body (otherwise use 'x' and '*' to fill) [Default=FALSE]
-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 (holdingconnections 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]
 --sequential                  Use sequential access (instead of random) [Default=FALSE]
 --start-at                    For sequential access, set the first item [Default=0]
 --rate-limit                  Set operations per second limit (per thread) [Default=0]
 --docs             User documents to load (overrides --min-size and --max-size [Default=]
 -J  --json                        Enable writing JSON values (rather than bytes) [Default=FALSE]
 --subdoc                      Use subdoc instead of fulldoc operations [Default=FALSE]
 --noop                        Use NOOP instead of document operations [Default=FALSE]
 --pathcount                   Number of subdoc paths per command [Default=1]
 --populate-only               Exit after documents have been populated [Default=FALSE]
 -e  --expiry                      Set TTL for items [Default=0]
 --persist-to     Wait until item is persisted to this number of nodes (-1 for master+replicas)
[Default=0]
--replicate-to    Wait until item is replicated to this number of nodes (-1 for all replicas)
[Default=0]
 --lock          Lock keys for updates for given time (will not lock when set to zero) [Default=0]
 -P  --password                    Bucket password [Default='']
 -u  --username                    Username [Default='']
 -Z  --config-cache                Path to cached configuration [Default='']
 -U  --spec                        Connection string [Default='couchbase://localhost/default']
 --truststorepath               [Default='']
 --certpath                    Path to server SSL certificate [Default='']
 --keypath                     Path to client SSL private key [Default='']
 -T  --timings  Enable command timings (second time todump timings automatically) [Default=FALSE]
 -v  --verbose   Set debugging output (specify multipletimes forgreater verbosity [Default=FALSE]
 --dump          Dump verbose internal state after operations are done [Default=FALSE]
 -y  --compress    Turn oncompression of outgoingdata (second time to force compression)
[Default=FALSE]
-D  --cparam  <OPTION=VALUE>      Additional options for connection. Use -Dtimeout=SECONDS for KV
operation timeout [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 Administrator -P
couchbase -U couchbase://$NODE1/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.**



beer-sample	7,303	100%	0	14.5MB / 100MB	20.1MB	Documents Statistics
default	510,000	8.54%	0	63.8MB / 100MB	122MB	Documents Statistics
gamesim-sample	586	100%	0	11.2MB / 100MB	10.9MB	Documents Statistics
travel-sample	31,591	100%	0	52.1MB / 100MB	65.2MB	Documents Statistics

Historical……

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

You can also use the cbc command to write a config file with cluster address and username and password set as cbc environmental variables in a .cbcrc file

Run the following command to set you clusterpath and username and password variables:

```
[ec2-user@ AppServer ~]$ cbc write-config -u Administrator -P
couchbase -U couchbase://$NODE1/default
```

Now check the file contents

```
[ec2-user@ AppServer ~]$ls -alh
total 346M
drwx------. 3 ec2-user ec2-user  227 Mar 18 20:00 .
drwxr-xr-x. 3 root     root       22 Mar 11 16:26 ..
-rw-rw-r--. 1 ec2-user ec2-user    0 Mar 14 18:56 ]
-rw-------. 1 ec2-user ec2-user 2.9K Mar 18 17:49 .bash_history
-rw-r--r--. 1 ec2-user ec2-user   18 Sep 26  2017 .bash_logout
-rw-r--r--. 1 ec2-user ec2-user  809 Mar 18 17:48 .bash_profile
-rw-r--r--. 1 ec2-user ec2-user  231 Sep 26  2017 .bashrc
-rw-rw-r--. 1 ec2-user ec2-user  187 Mar 18 20:01 .cbcrc
-rw-rw-r--. 1 ec2-user ec2-user 345M Feb  1 17:31 couchbase-server-
enterprise-6.5.1-centos7.x86_64.rpm
-rw-rw-r--. 1 ec2-user ec2-user 328K Mar 18 19:42 jsaon_output_file
-rw-rw-r--. 1 ec2-user ec2-user 328K Mar 18 19:42 json_output_file
drwx------. 2 ec2-user ec2-user   29 Mar 11 16:26 .ssh
```

```
[ec2-user@appserver ~]$ cat .cbcrc

# Generated by cbc at Mon Mar 18 20:01:52 2019
connstr=couchbase://ec2-13-56-188-91.us-west-
1.compute.amazonaws.com/default?username=Administrator&
user=Administrator
password=couchbase
```

```
[ec2-user@AppServer ~]$ cbc-create --help
Usage:
  create [OPTIONS...] KEY -V VALUE

Store item to the server
  -P  --password                 Bucket password [Default='couchbase']
  -u  --username                 Username [Default='Administrator']
  -Z  --config-cache             Path to cached configuration [Default='']
  -U  --spec                     Connection string [Default='couchbase://ec2-18-236-106-89.us-west-
2.compute.amazonaws.com/default?username=Administrator&']
      --truststorepath           [Default='']
      --certpath                 Path to server SSL certificate [Default='']
      --keypath                  Path to client SSL private key [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]
  -y  --compress                 Turn on compression of outgoing data (second time to force
compression) [Default=FALSE]
  -D  --cparam  <OPTION=VALUE>   Additional options for connection. Use -
Dtimeout=SECONDS for KV operation timeout [Default=]
  -M  --mode  <upsert|insert|replace> Mode to use when storing [Default='upsert']
  -f  --flags                    Flags for item [Default=0]
  -e  --expiry                   Expiry for item [Default=0]
  -p  --persist-to               Wait until item is persisted to this number of nodes [Default=0]
  -r  --replicate-to             Wait until item is replicated to this number of nodes [Default=0]
  -V  --value                    Value to use. If unspecified, read from standard input [Default='']
  -J  --json                     Indicate to the server that this item is JSON [Default=FALSE]
  -?  --help                     this message
```

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 -f 555 -V this_is_my_VALUE_in_ascii
cbc_key
cbc_key                 Stored. CAS=0x15ec1965db640000
```

Retrieve the cbc_key:

```
[ec2-user@ AppServer ~]$ cbc-cat cbc_key
cbc_key                 CAS=0x15ec1a0a53570000, Flags=0x22b, Size=25,
Datatype=0x00
this_is_my_VALUE_in_ascii
```

Delete the cbc_key:

```
[ec2-user@ AppServer ~]$ cbc-rm cbc_key

cbc_key             Deleted. CAS=0x1548adc55c250000
```

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

This concludes lab #2.0