**THIS DOCUMENT IS OBSOLETE**

**Mongo Install Instructions:**

All NOSQL s/w solution packages can be configured to run on a single node or a cluster. A single node can be called pseudo-distributed mode where multiple processes are run on one machine to simulate running on a cluster. Each process on a single machine is the replacement of running the process on a separate server/node/IP Address.

**Single Node Mongo DB instructions:**

1. Assume a CentOS linux distribution, follow the instructions:

<http://docs.mongodb.org/manual/tutorial/install-mongodb-on-redhat-centos-or-fedora-linux/>

2)

**Cluster install.**

A mongodDB cluster is also called a sharded MongoDB cluster. Mongodb terminology is to use the term sharding to refer to a clustered MongoDB configuration. The instructions for creating a MongoDB cluster are here: <http://www.mongodb.org/display/DOCS/A+Sample+Configuration+Session>

A mongod cluster is defined by

1. multiple mongod shard servers, 1 or 3 required
2. A config server
3. A mongos process which is the master

First start the mongod servers. For this example we will use 2 servers. Each of these mongod servers store the data which is written to the master or mongos server.

**Start a mongod process, call this mongo server a.**

**>mongod --shardsvr --dbpath /data/db/a --port 10000 > /tmp/sharda.log &**

**After starting the server, verify the log file is correct, it should look like the output below where the journal file is preallocatd.**

[ec2-user@ip-10-28-173-121 ~]$ cat /tmp/sharda.log

Tue Aug 21 17:46:54 [initandlisten] MongoDB starting : pid=2610 port=10000 dbpath=/data/db/a 64-bit host=ip-10-28-173-121

Tue Aug 21 17:46:54 [initandlisten] db version v2.0.7, pdfile version 4.5

Tue Aug 21 17:46:54 [initandlisten] git version: 875033920e8869d284f32119413543fa475227bf

Tue Aug 21 17:46:54 [initandlisten] build info: Linux ip-10-2-29-40 2.6.21.7-2.ec2.v1.2.fc8xen #1 SMP Fri Nov 20 17:48:28 EST 2009 x86\_64 BOOST\_LIB\_VERSION=1\_41

Tue Aug 21 17:46:54 [initandlisten] options: { dbpath: "/data/db/a", port: 10000, shardsvr: true }

Tue Aug 21 17:46:54 [initandlisten] journal dir=/data/db/a/journal

Tue Aug 21 17:46:54 [initandlisten] recover begin

Tue Aug 21 17:46:54 [initandlisten] info no lsn file in journal/ directory

Tue Aug 21 17:46:54 [initandlisten] recover lsn: 0

Tue Aug 21 17:46:54 [initandlisten] recover /data/db/a/journal/j.\_0

Tue Aug 21 17:46:54 [initandlisten] recover cleaning up

Tue Aug 21 17:46:54 [initandlisten] removeJournalFiles

Tue Aug 21 17:46:54 [initandlisten] recover done

Tue Aug 21 17:46:54 [initandlisten] preallocating a journal file /data/db/a/journal/prealloc.0

775946240/1073741824 72%

880803840/1073741824 82%

891289600/1073741824 83%

Tue Aug 21 17:47:29 [initandlisten] waiting for connections on port 10000

Tue Aug 21 17:47:29 [websvr] admin web console waiting for connections on port 11000

Tue Aug 21 17:48:29 [clientcursormon] mem (MB) res:14 virt:628 mapped:0

**Start the second mongod server, server b.**

**mongod --shardsvr --dbpath /data/db/b --port 10000 > /tmp/shardb.log &**

**Verify the logs show mongodb is running correctly on server b:**

[ec2-user@ip-10-28-173-121 ~]$ cat /tmp/shardb.log

Tue Aug 21 17:47:31 [initandlisten] MongoDB starting : pid=2621 port=10001 dbpath=/data/db/b 64-bit host=ip-10-28-173-121

Tue Aug 21 17:47:31 [initandlisten] db version v2.0.7, pdfile version 4.5

Tue Aug 21 17:47:31 [initandlisten] git version: 875033920e8869d284f32119413543fa475227bf

Tue Aug 21 17:47:31 [initandlisten] build info: Linux ip-10-2-29-40 2.6.21.7-2.ec2.v1.2.fc8xen #1 SMP Fri Nov 20 17:48:28 EST 2009 x86\_64 BOOST\_LIB\_VERSION=1\_41

Tue Aug 21 17:47:31 [initandlisten] options: { dbpath: "/data/db/b", port: 10001, shardsvr: true }

Tue Aug 21 17:47:31 [initandlisten] journal dir=/data/db/b/journal

Tue Aug 21 17:47:31 [initandlisten] recover begin

Tue Aug 21 17:47:31 [initandlisten] info no lsn file in journal/ directory

Tue Aug 21 17:47:31 [initandlisten] recover lsn: 0

Tue Aug 21 17:47:31 [initandlisten] recover /data/db/b/journal/j.\_0

Tue Aug 21 17:47:31 [initandlisten] recover cleaning up

Tue Aug 21 17:47:31 [initandlisten] removeJournalFiles

Tue Aug 21 17:47:31 [initandlisten] recover done

Tue Aug 21 17:47:31 [initandlisten] preallocating a journal file /data/db/b/journal/prealloc.0

796917760/1073741824 74%

901775360/1073741824 83%

912261120/1073741824 84%

Tue Aug 21 17:48:04 [initandlisten] waiting for connections on port 10001

Tue Aug 21 17:48:04 [websvr] admin web console waiting for connections on port 11001

Tue Aug 21 17:49:04 [clientcursormon] mem (MB) res:14 virt:628 mapped:0

**Now start the config servers and sharding servers:**

[ec2-user@ip-10-28-173-121 ~]$ mongod --configsvr --dbpath /data/db/config --port 20000 > /tmp/config.log &

[3] 2634

[ec2-user@ip-10-28-173-121 ~]$ cat /tmp/config.log

Tue Aug 21 17:49:06 [initandlisten] MongoDB starting : pid=2635 port=20000 dbpath=/data/db/config 64-bit host=ip-10-28-173-121

Tue Aug 21 17:49:06 [initandlisten] db version v2.0.7, pdfile version 4.5

Tue Aug 21 17:49:06 [initandlisten] git version: 875033920e8869d284f32119413543fa475227bf

Tue Aug 21 17:49:06 [initandlisten] build info: Linux ip-10-2-29-40 2.6.21.7-2.ec2.v1.2.fc8xen #1 SMP Fri Nov 20 17:48:28 EST 2009 x86\_64 BOOST\_LIB\_VERSION=1\_41

Tue Aug 21 17:49:06 [initandlisten] options: { configsvr: true, dbpath: "/data/db/config", port: 20000 }

Tue Aug 21 17:49:06 [initandlisten] journal dir=/data/db/config/journal

Tue Aug 21 17:49:06 [initandlisten] recover begin

Tue Aug 21 17:49:06 [initandlisten] info no lsn file in journal/ directory

Tue Aug 21 17:49:06 [initandlisten] recover lsn: 0

Tue Aug 21 17:49:06 [initandlisten] recover /data/db/config/journal/j.\_0

Tue Aug 21 17:49:06 [initandlisten] recover cleaning up

Tue Aug 21 17:49:06 [initandlisten] removeJournalFiles

Tue Aug 21 17:49:06 [initandlisten] recover done

Tue Aug 21 17:49:06 [initandlisten] preallocating a journal file /data/db/config/journal/prealloc.0

754974720/1073741824 70%

**[ec2-user@ip-10-28-173-121 ~]$ mongos --configdb localhost:20000 > /tmp/mongos.log &**

[4] 2647

[ec2-user@ip-10-28-173-121 ~]$ cat /tmp/mongos.log

Tue Aug 21 17:49:54 mongos db version v2.0.7, pdfile version 4.5 starting (--help for usage)

Tue Aug 21 17:49:54 git version: 875033920e8869d284f32119413543fa475227bf

Tue Aug 21 17:49:54 build info: Linux ip-10-2-29-40 2.6.21.7-2.ec2.v1.2.fc8xen #1 SMP Fri Nov 20 17:48:28 EST 2009 x86\_64 BOOST\_LIB\_VERSION=1\_41

Tue Aug 21 17:49:54 [websvr] admin web console waiting for connections on port 28017

Tue Aug 21 17:49:54 [mongosMain] waiting for connections on port 27017

Tue Aug 21 17:49:54 [Balancer] about to contact config servers and shards

Tue Aug 21 17:49:54 [Balancer] config servers and shards contacted successfully

Tue Aug 21 17:49:54 [Balancer] balancer id: ip-10-28-173-121:27017 started at Aug 21 17:49:54

Tue Aug 21 17:49:54 [Balancer] creating WriteBackListener for: localhost:20000 serverID: 5033ca4236e0d17be2ed09e2

Tue Aug 21 17:49:54 [LockPinger] creating distributed lock ping thread for localhost:20000 and process ip-10-28-173-121:27017:1345571394:1804289383 (sleeping for 30000ms)

Tue Aug 21 17:49:54 [Balancer] distributed lock 'balancer/ip-10-28-173-121:27017:1345571394:1804289383' acquired, ts : 5033ca4236e0d17be2ed09e3

Tue Aug 21 17:49:54 [Balancer] distributed lock 'balancer/ip-10-28-173-121:27017:1345571394:1804289383' unlocked.

[ec2-user@ip-10-28-173-121 ~]$

To verify everything is started correctly, you should see 4 mongo\* processes, 2 mongod servers, one mongo config server and one mongos server:

[ec2-user@ip-10-28-173-121 ~]$ ps -ef | grep mongo

root 2610 1 0 17:46 pts/1 00:00:01 mongod -shardsvr --dbpath /data/db/a --port 10000

root 2621 1 0 17:47 pts/1 00:00:01 mongod -shardsvr --dbpath /data/db/b --port 10001

root 2635 1 0 17:49 pts/1 00:00:01 mongod --configsvr --dbpath /data/db/config --port 20000

ec2-user 2792 1594 0 18:15 pts/1 00:00:00 mongos --configdb localhost:20000 --chunkSize 1

ec2-user 2809 1594 0 18:15 pts/1 00:00:00 grep mongo

[ec2-user@ip-10-28-173-121 ~]$

**Verify you have the command line mongo client working:**

[ec2-user@ip-10-28-173-121 ~]$ mongo

MongoDB shell version: 2.0.7

connecting to: test

mongos>

**Add the shards:**

mongos> use admin

switched to db admin

mongos> db.runCommand({addshard:"localhost:10000"})

{ "shardAdded" : "shard0000", "ok" : 1 }

mongos> db.runCommand({addshard:"localhost:10001"})

{ "shardAdded" : "shard0001", "ok" : 1 }

mongos>

**Verify the shards are working:**

mongos> use admin

switched to db admin

mongos> db.runCommand({listShards:1})

{

"shards" : [

{

"\_id" : "shard0000",

"host" : "localhost:10000"

},

{

"\_id" : "shard0001",

"host" : "localhost:10001"

}

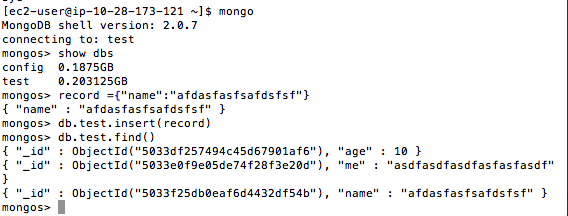
],

"ok" : 1

}

mongos>

**Verify mongo in cluster mode, you should see the mongos prompt:**

****

**PyMongo Examples:**

To test pymongo is operating correctly, verify you can connect to a database and display the count in the collection. This is the same syntax as using the mongo command line

Inserting data using pymongo:

>>>import pymongo

>>> connection=pymongo.Connection()

>>> testrecord={"name":"testpython"}

>>> db.test.insert(testrecord)

ObjectId('5033f3e0c995ab0cbe5798d9')

>>> db.test.count()

4

>>>

One of the advantages of using the python command line is the ability to do inserts without compiling a program or running from inside an IDE:

>>> for x in range(10):

... db.test.insert({"name":x})

...

ObjectId('5033f656c995ab0cbe5798da')

ObjectId('5033f656c995ab0cbe5798db')

ObjectId('5033f656c995ab0cbe5798dc')

ObjectId('5033f656c995ab0cbe5798dd')

ObjectId('5033f656c995ab0cbe5798de')

ObjectId('5033f656c995ab0cbe5798df')

ObjectId('5033f656c995ab0cbe5798e0')

ObjectId('5033f656c995ab0cbe5798e1')

ObjectId('5033f656c995ab0cbe5798e2')

ObjectId('5033f656c995ab0cbe5798e3')

>>> db.test.count()

14

**Python Unicode strings: strings are denoted using the lower case u in front of the string. In the example below we are connecting to the collection or table test which needs a u in front of it.**

>>> print db

Database(Connection('localhost', 27017), u'test')

>>>

**Avoiding single points of failure: Sharding and Replication**

A properly configured mongodb cluster has no single point of failure. Here is an example cluster setup using AWS/EBS.

NO referential integrity, can’t test to make sure you have certain type in the mongodb field.

**DB.posts.find().explain()**

Cover BTreeCursor vs. BasicCursor

Cover padding

Tagging: add? Nodes in other clusters? Not necessary.

**Typical Error messages:**

**can't create user databases on a --configsvr instance**

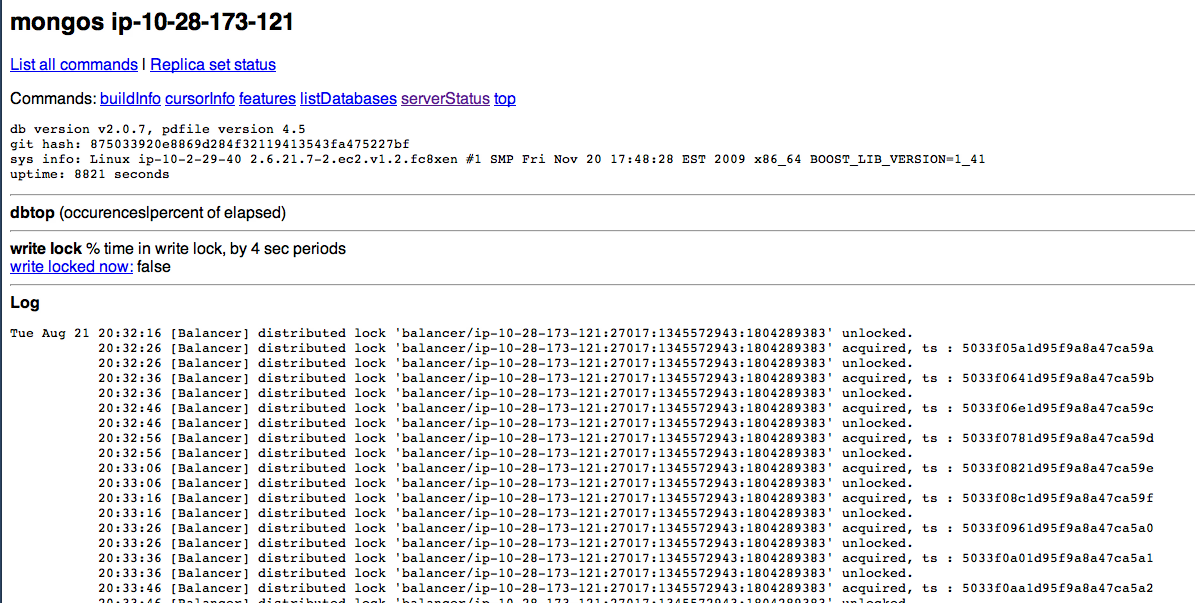
Connected to wrong mongodb instance. Connect to the mongos instance, not the config server.

**Verify Mongo installed correctly**

Logs through WEB API: Add url + 27017 in the port

ec2-23-21-22-237.compute-1.amazonaws.com

:27017

****