**MongoDB One Server Install and Config**

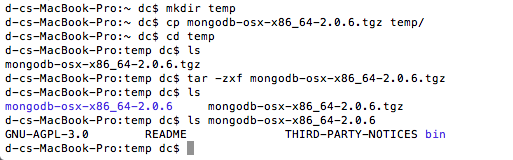
The starting point for MongoDB is to install MongoDB on a single machine and run it in 2 modes, single mongod process mode and then 3 mongod process mode which simulates a cluster on a single machine. We can call this second mode pseudo-distrubuted mode.

This is the simplest case and removes potential complications from file and directory permissions and having to get access to a separate server.

Download and install mongodb. You don’t have to use the MongoDB Yum/APT repos for this handout. Feel free to use a Mac or Windows or linux laptop/machine.

These instructions show screenshots from a Mac:

After install the mongodb directory should look like below where there is a subdirectory bin/ with the mongodb executables:



MongoDB executables:

Macintosh HD:Users:dc:Desktop:Screen Shot 2012-09-01 at 2.37.37 AM.png

We will cover all the programs in red as they are useful in maintaining the runtime and debugging production problems with the cluster and data.

For MongoDB, there is no documented recovery process for either undoing a transaction a user executed or a recovery from cluster crashes. We have to simulate and develop these for each use case.

First lets make sure the install works by starting a single mongodb process which represents the database and a mongo command line client to read and write data into and out of the database.

This is equivalent to starting a SQL database and a SQL Management Studio GUI which allows us to create/drop tables, set permissions, create indexes, and read/write data into tables in the db.

Before starting the server in a single node we have to establish some conventions. In our case we will store the db data into a directory named slaveX/ where X is the number of slaves we have in a cluster and the db files are stored in a directory named db and log files in a log file under slaveX.

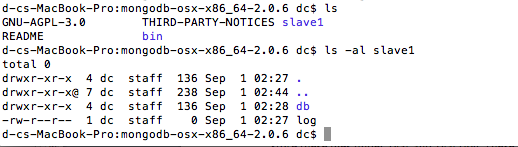
Linux has certain conventions which are predefined across the industry and will store these files under /var and /var/log. These may differ by distribution. We will cover linux specific conventions in a separate document.

Under the mongodb install directory, create a directory slave1/db and a log file /slave1/log

>mkdir –p /slave1/db

>touch /slave1/log

Your directory should look like this:



Start the mongodb process. Specify 3 parameters, the path to the db, the path to the log file and the port number. The default is 27017. You can leave the port number option and log file and db path off. Mongod will use the defaults in mongod.conf.

>bin/mongod –dbpath (path to directory)/slave1/db –logpath (path to directory)slave1/log –port 10000

Example:

d-cs-MacBook-Pro:mongodb-osx-x86\_64-2.0.6 dc$ bin/mongod --dbpath /Users/dc/mongodb-osx-x86\_64-2.0.6/slave1/db --logpath /Users/dc/mongodb-osx-x86\_64-2.0.6/slave1/log --port 10000

Start the client, since we started mongod on localhost port 10000 we have to specify this or the client looks for the default port 27017. The host parameter isn’t necessary but when we start dealing with clusters this will become important and these simple startup parameters are a big source of user error from configuration.

>mongo –host localhost –port 10000

>db.test.add({“a”:”asdf”})

show the existing dbs

>show dbs

Example:

d-cs-MacBook-Pro:mongodb-osx-x86\_64-2.0.6 dc$ bin/mongo --host localhost --port 10000

MongoDB shell version: 2.0.6

connecting to: localhost:10000/test

> use admin

switched to db admin

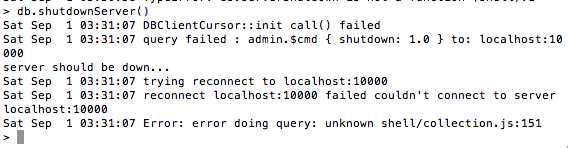
> show dbs

local (empty)

test 0.203125GB

> db.serverShutdown()

Note: the docs say you aren’t supposed to kill the process but do a server shutdown from the command line client which is demonstrated above. As a test this gives an error message as seen below. The error message seems incorrect;



After shutdown you can see the process terminate but you see an error message from the client:



After verifying this works in single process mode, test pseudo-distributed mode. We simulate this by starting 3 mongod database processes, 1 mongod config server process and 1 mongos process.

One of the changes we are going to make is we are going to either run the monogd process in the background or fork off the mongod process. Fork off 3 mongod processes in ports 10000, 12000, 14000

d-cs-MacBook-Pro:mongodb-osx-x86\_64-2.0.6 dc$ bin/mongod --dbpath /Users/dc/mongodb-osx-x86\_64-2.0.6/slave1/db --logpath /Users/dc/mongodb-osx-x86\_64-2.0.6/slave1/log --port 10000 --fork

d-cs-MacBook-Pro:mongodb-osx-x86\_64-2.0.6 dc$ bin/mongod --dbpath /Users/dc/mongodb-osx-x86\_64-2.0.6/slave1/db --logpath /Users/dc/mongodb-osx-x86\_64-2.0.6/slave1/log --port 12000 --fork

forked process: 61845

d-cs-MacBook-Pro:mongodb-osx-x86\_64-2.0.6 dc$ all output going to: /Users/dc/mongodb-osx-x86\_64-2.0.6/slave1/log

d-cs-MacBook-Pro:mongodb-osx-x86\_64-2.0.6 dc$ bin/mongod --dbpath /Users/dc/mongodb-osx-x86\_64-2.0.6/slave1/db --logpath /Users/dc/mongodb-osx-x86\_64-2.0.6/slave1/log --port 14000 --fork

forked process: 61849

d-cs-MacBook-Pro:mongodb-osx-x86\_64-2.0.6 dc$ all output going to: /Users/dc/mongodb-osx-x86\_64-2.0.6/slave1/log

We should be able to migrate the database in slave1 to a shared form where it replicates the data 3 times and we should be able to track these multiple copies.