**\*\*\*\*\*\*\*HOW TO FIND THE PROCESS SO IT CAN BE KILLED\*\*\*\*\*\*\***

vagrant@precise64:~/mongotraining$ ps -Aef | grep mongod

**Steps to configure the server for 4.2**

**Step 1:**

vagrant@precise64:~/mongotraining$ mongo --port 27001 --shell week4.js

MongoDB shell version: 2.4.5

connecting to: 127.0.0.1:27001/test

type "help" for help

**Step 2:**

vagrant@precise64:~/mongotraining$ mongod --dbpath 1 --port 27001 --smallfiles --oplogSize 50 --replSet abc --logpath log.1 --fork

about to fork child process, waiting until server is ready for connections.

forked process: 1978

all output going to: /home/vagrant/mongotraining/log.1

log file [/home/vagrant/mongotraining/log.1] exists; copied to temporary file [/home/vagrant/mongotraining/log.1.2013-08-09T16-40-38]

child process started successfully, parent exiting

vagrant@precise64:~/mongotraining$ mongod --dbpath 2 --port 27002 --smallfiles --oplogSize 50 --replSet abc --logpath log.2 --fork

about to fork child process, waiting until server is ready for connections.

forked process: 2024

all output going to: /home/vagrant/mongotraining/log.2

child process started successfully, parent exiting

vagrant@precise64:~/mongotraining$ mongod --dbpath 3 --port 27003 --smallfiles --oplogSize 50 --replSet abc --logpath log.3 --fork

about to fork child process, waiting until server is ready for connections.

forked process: 2070

all output going to: /home/vagrant/mongotraining/log.3

child process started successfully, parent exiting

**Step 3:**

cfg = {

\_id : "abc",

members : [

{ \_id:0, host:"precise64:27001"},

{ \_id:1, host:"precise64:27002"},

{ \_id:2, host:"precise64:27003"}

]

}

**Step 4:**

> rs.initiate()

{

"info2" : "no configuration explicitly specified -- making one",

"me" : "precise64:27001",

"info" : "Config now saved locally. Should come online in about a minute.",

"ok" : 1

}

**Solution to 4.2**

vagrant@precise64:~/mongotraining$ mongo --port 27001 --shell week4.js

MongoDB shell version: 2.4.5

connecting to: 127.0.0.1:27001/test

type "help" for help

abc:PRIMARY> use week4

switched to db week4

abc:PRIMARY> db.foo.find()

{ "\_id" : ObjectId("520518a698c7d1a123b008ca"), "x" : 0, "y" : 0.8544979079160839 }

{ "\_id" : ObjectId("520518a698c7d1a123b008cb"), "x" : 1, "y" : 0.07335674343630672 }

{ "\_id" : ObjectId("520518a698c7d1a123b008cc"), "x" : 2, "y" : 0.9171984707936645 }

{ "\_id" : ObjectId("520518a698c7d1a123b008cd"), "x" : 3, "y" : 0.39636226440779865 }

{ "\_id" : ObjectId("520518a698c7d1a123b008ce"), "x" : 4, "y" : 0.03707383759319782 }

{ "\_id" : ObjectId("520518a698c7d1a123b008cf"), "x" : 5, "y" : 0.16457181912846863 }

{ "\_id" : ObjectId("520518a698c7d1a123b008d0"), "x" : 6, "y" : 0.4285534133668989 }

{ "\_id" : ObjectId("520518a698c7d1a123b008d1"), "x" : 7, "y" : 0.1455568776000291 }

{ "\_id" : ObjectId("520518a698c7d1a123b008d2"), "x" : 8, "y" : 0.4020087090320885 }

{ "\_id" : ObjectId("520518a698c7d1a123b008d3"), "x" : 9, "y" : 0.056489973329007626 }

{ "\_id" : ObjectId("520518a698c7d1a123b008d4"), "x" : 10, "y" : 0.4118536196183413 }

{ "\_id" : ObjectId("520518a698c7d1a123b008d5"), "x" : 11, "y" : 0.6638806867413223 }

{ "\_id" : ObjectId("520518a698c7d1a123b008d6"), "x" : 12, "y" : 0.4041961939074099 }

{ "\_id" : ObjectId("520518a698c7d1a123b008d7"), "x" : 13, "y" : 0.21224260027520359 }

{ "\_id" : ObjectId("520518a698c7d1a123b008d8"), "x" : 14, "y" : 0.3674898617900908 }

{ "\_id" : ObjectId("520518a698c7d1a123b008d9"), "x" : 15, "y" : 0.7240571700967848 }

{ "\_id" : ObjectId("520518a698c7d1a123b008da"), "x" : 16, "y" : 0.23662885301746428 }

{ "\_id" : ObjectId("520518a698c7d1a123b008db"), "x" : 17, "y" : 0.7142833324614912 }

{ "\_id" : ObjectId("520518a698c7d1a123b008dc"), "x" : 18, "y" : 0.6832155333831906 }

{ "\_id" : ObjectId("520518a698c7d1a123b008dd"), "x" : 19, "y" : 0.8521488995756954 }

Type "it" for more

abc:PRIMARY> homework.b()

5002

References:

<http://www.kchodorow.com/blog/2012/03/06/replica-set-internals-bootcamp-part-ii-creating-a-set/>

# Question 4.3

**On server shell 2 (mongod shell):**

vagrant@precise64:~/mongotraining$ mongod --dbpath 1 --port 27001 --smallfiles --oplogSize 50 --replSet abc --logpath log.1 --fork

about to fork child process, waiting until server is ready for connections.

forked process: 1416

all output going to: /home/vagrant/mongotraining/log.1

child process started successfully, parent exiting

vagrant@precise64:~/mongotraining$ mongod --dbpath 2 --port 27002 --smallfiles --oplogSize 50 --replSet abc --logpath log.2 --fork

about to fork child process, waiting until server is ready for connections.

forked process: 1462

all output going to: /home/vagrant/mongotraining/log.2

child process started successfully, parent exiting

vagrant@precise64:~/mongotraining$ mongod --dbpath 3 --port 27003 --smallfiles --oplogSize 50 --replSet abc --logpath log.3 --fork

about to fork child process, waiting until server is ready for connections.

forked process: 1508

all output going to: /home/vagrant/mongotraining/log.3

child process started successfully, parent exiting

**On server shell 1 (mongo shell):**

vagrant@precise64:~/mongotraining$ mongo --port 27001 --shell week4.js

MongoDB shell version: 2.4.5

connecting to: 127.0.0.1:27001/test

type "help" for help

> rs.status()

{

"startupStatus" : 3,

"info" : "run rs.initiate(...) if not yet done for the set",

"ok" : 0,

"errmsg" : "can't get local.system.replset config from self or any seed (EMPTYCONFIG)"

}

> cfg = {

... \_id : "abc",

... members : [

... { \_id:0, host:"precise64:27001"},

... { \_id:1, host:"precise64:27002"},

... { \_id:2, host:"precise64:27003"}

... ]

... }

{

"\_id" : "abc",

"members" : [

{

"\_id" : 0,

"host" : "precise64:27001"

},

{

"\_id" : 1,

"host" : "precise64:27002"

},

{

"\_id" : 2,

"host" : "precise64:27003"

}

]

}

> rs.initiate(cfg)

{

"info" : "Config now saved locally. Should come online in about a minute.",

"ok" : 1

}

> ^C

bye

**Verify that each server is in the proper role (i.e. Primary or Secondary)**

vagrant@precise64:~/mongotraining$ mongo --port 27001 --shell week4.js

MongoDB shell version: 2.4.5

connecting to: 127.0.0.1:27001/test

type "help" for help

abc:PRIMARY> exit

bye

vagrant@precise64:~/mongotraining$ mongo --port 27002 --shell week4.js

MongoDB shell version: 2.4.5

connecting to: 127.0.0.1:27002/test

type "help" for help

abc:SECONDARY> exit

bye

vagrant@precise64:~/mongotraining$ mongo --port 27003 --shell week4.js

MongoDB shell version: 2.4.5

connecting to: 127.0.0.1:27003/test

type "help" for help

abc:SECONDARY> exit

bye

abc:SECONDARY> rs.status()

{

"set" : "abc",

"date" : ISODate("2013-08-12T13:33:30Z"),

"myState" : 2,

"syncingTo" : "precise64:27001",

"members" : [

{

"\_id" : 0,

"name" : "precise64:27001",

"health" : 1,

"state" : 1,

"stateStr" : "PRIMARY",

"uptime" : 668,

"optime" : Timestamp(1376313728, 1),

"optimeDate" : ISODate("2013-08-12T13:22:08Z"),

"lastHeartbeat" : ISODate("2013-08-12T13:33:28Z"),

"lastHeartbeatRecv" : ISODate("2013-08-12T13:33:29Z"),

"pingMs" : 0

},

{

"\_id" : 1,

"name" : "precise64:27002",

"health" : 1,

"state" : 2,

"stateStr" : "SECONDARY",

"uptime" : 763,

"optime" : Timestamp(1376313728, 1),

"optimeDate" : ISODate("2013-08-12T13:22:08Z"),

"self" : true

},

{

"\_id" : 2,

"name" : "precise64:27003",

"health" : 1,

"state" : 2,

"stateStr" : "SECONDARY",

"uptime" : 668,

"optime" : Timestamp(1376313728, 1),

"optimeDate" : ISODate("2013-08-12T13:22:08Z"),

"lastHeartbeat" : ISODate("2013-08-12T13:33:28Z"),

"lastHeartbeatRecv" : ISODate("2013-08-12T13:33:30Z"),

"pingMs" : 0,

"syncingTo" : "precise64:27001"

}

],

"ok" : 1

}

vagrant@precise64:~/mongotraining$ mongo --port 27002 week4.js --shell

MongoDB shell version: 2.4.5

connecting to: 127.0.0.1:27002/test

type "help" for help

abc:SECONDARY> db.isMaster()

{

"setName" : "abc",

"ismaster" : false,

"secondary" : true,

"hosts" : [

"precise64:27002",

"precise64:27003",

"precise64:27001"

],

"primary" : "precise64:27001",

"me" : "precise64:27002",

"maxBsonObjectSize" : 16777216,

"maxMessageSizeBytes" : 48000000,

"localTime" : ISODate("2013-08-12T13:37:25.425Z"),

"ok" : 1

}

**Solution to 4.3**

vagrant@precise64:~/mongotraining$ mongo --port 27002 week4.js --shell

MongoDB shell version: 2.4.5

connecting to: 127.0.0.1:27002/test

type "help" for help

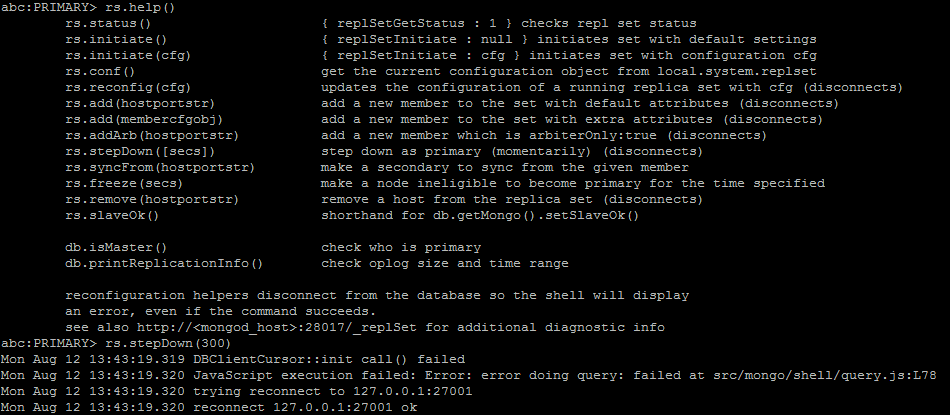
abc:SECONDARY> homework.c()

5

**Question 4.4**

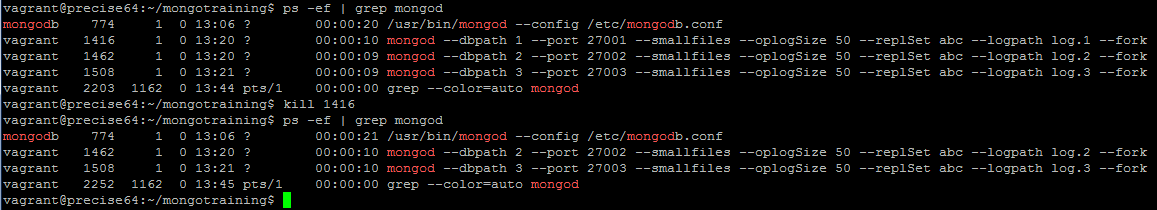
Part 1 (on mongo shell):

***rs.stepdown(300)*** causes the primary @ 27001 to step down to secondary and an election will be held for the new primary.



Part 2 (on mongod shell):

grep the mongod process and kill the one for 27001(old primary) to shut it down.



This will cause the 01 to go offline, and leave 27002 and 27003 online. A new primary will be elected from this group.

Connect to the 27002 server via $ mongo --port 27002 week4.js –shell

Use $ rs.status() to check the status of the replica set and its individuals.

vagrant@precise64:~/mongotraining$ mongo --port 27001 week4.js --shell

MongoDB shell version: 2.4.5

connecting to: 127.0.0.1:27001/test

Mon Aug 12 13:46:43.432 JavaScript execution failed: Error: couldn't connect to server 127.0.0.1:27001 at src/mongo/shell/mongo.js:L114

exception: connect failed

vagrant@precise64:~/mongotraining$ mongo --port 27002 week4.js --shell

MongoDB shell version: 2.4.5

connecting to: 127.0.0.1:27002/test

type "help" for help

abc:PRIMARY> rs.status()

{

"set" : "abc",

"date" : ISODate("2013-08-12T13:46:53Z"),

"myState" : 1,

"members" : [

{

"\_id" : 0,

"name" : "precise64:27001",

"health" : 0,

"state" : 8,

"stateStr" : "(not reachable/healthy)",

"uptime" : 0,

"optime" : Timestamp(1376313728, 1),

"optimeDate" : ISODate("2013-08-12T13:22:08Z"),

"lastHeartbeat" : ISODate("2013-08-12T13:46:53Z"),

"lastHeartbeatRecv" : ISODate("2013-08-12T13:45:44Z"),

"pingMs" : 0,

"syncingTo" : "precise64:27002"

},

{

"\_id" : 1,

"name" : "precise64:27002",

"health" : 1,

"state" : 1,

"stateStr" : "PRIMARY",

"uptime" : 1566,

"optime" : Timestamp(1376313728, 1),

"optimeDate" : ISODate("2013-08-12T13:22:08Z"),

"self" : true

},

{

"\_id" : 2,

"name" : "precise64:27003",

"health" : 1,

"state" : 2,

"stateStr" : "SECONDARY",

"uptime" : 1471,

"optime" : Timestamp(1376313728, 1),

"optimeDate" : ISODate("2013-08-12T13:22:08Z"),

"lastHeartbeat" : ISODate("2013-08-12T13:46:53Z"),

"lastHeartbeatRecv" : ISODate("2013-08-12T13:46:53Z"),

"pingMs" : 0,

"syncingTo" : "precise64:27002"

}

],

"ok" : 1

}

We can connect to 27002 now, and we see that it is the primary

vagrant@precise64:~/mongotraining$ mongo --port 27002 week4.js --shell

MongoDB shell version: 2.4.5

connecting to: 127.0.0.1:27002/test

type "help" for help

abc:PRIMARY>

And now we want to remove 27001 from the replica set

abc:PRIMARY> rs.remove('precise64:27001')

Mon Aug 12 13:57:37.477 DBClientCursor::init call() failed

Mon Aug 12 13:57:37.477 JavaScript execution failed: Error: error doing query: failed at src/mongo/shell/query.js:L78

Mon Aug 12 13:57:37.478 trying reconnect to 127.0.0.1:27002

Mon Aug 12 13:57:37.478 reconnect 127.0.0.1:27002 ok

Check to see that 27001 has been removed:

abc:PRIMARY> db.isMaster()

{

"setName" : "abc",

"ismaster" : true,

"secondary" : false,

"hosts" : [

"precise64:27002",

"precise64:27003"

],

"primary" : "precise64:27002",

"me" : "precise64:27002",

"maxBsonObjectSize" : 16777216,

"maxMessageSizeBytes" : 48000000,

"localTime" : ISODate("2013-08-12T13:57:44.191Z"),

"ok" : 1

}

**Solution to 4.4**

abc:PRIMARY> homework.d()

6

**Question 4.5**

abc:SECONDARY> db.isMaster().ismaster

false

abc:SECONDARY> use local

switched to db local

abc:SECONDARY> db.oplog.rs.find()

{ "ts" : Timestamp(1376313728, 1), "h" : NumberLong(0), "v" : 2, "op" : "n", "ns" : "", "o" : { "msg" : "initiating set" } }

{ "ts" : Timestamp(1376315857, 1), "h" : NumberLong("8257216742732857362"), "v" : 2, "op" : "n", "ns" : "", "o" : { "msg" : "Reconfig set", "version" : 2 } }

I get the incorrect answer of “i”. Trying to figure out what happened….