{ Ops : Workshop }

MongoDB London
Sam Weaver
@samuel_weaver

mongoDB
{name: "mongo", type:"DB"}

Agenda

- Intro to MongoDB
- Replica Sets
- Sharding
- Other topics
- Hands on!
- Questions? Ask!

mongoDB

{name: "mongo", type:"DB"}

Getting Started

- Download mongodb
  - Windows, Linux, Mac OSX
- http://www.mongodb.org/downloads
- Odd and even version numbers?
- Unzip and go!

mongoDB
{name: "mongo", type:"DB"}

Running a single mongod

- Make a directory for your data

```
mkdir -p /data/db
```

- Start up mongod

```
mongod --dbpath /data/db
```

- --logpath, --logappend, --fork, --rest, --port

mongoDB
{name: "mongo", type:"DB"}

Connect and use

- mongo
- `show dbs`
- `show collections`
- `use <database>`
- `db.test.insert()`
- `db.test.find()`

mongoDB
{name: "mongo", type:"DB"}

Database files

- Check /data/db
- test.ns 16MB
- test.0 64MB
- test.1 128MB
- "Huh? What's up with all these files?"

mongoDB

{name: "mongo", type:"DB"}

Exercise

- Download and install mongodb
- Connect to the shell
- Create a new "workshop" database
- Create a document in the "test" collection

mongoDB
{name: "mongo", type:"DB"}

Replica Sets

- What are replica sets?
    - Quorum, heartbeat
- Primary and secondary’s
    - Replication
    - Oplog & oplog sizing
- Automatic Failover
    - Voting
- Write concern

mongoDB

{name: “mongo”, type:“DB”}

Setting up a replica set

- make a data directory for each node

```
mongod --dbpath /data/node1 --replSet test --port 27017
mongod --dbpath /data/node2 --replSet test --port 27018
mongod --dbpath /data/node3 --replSet test --port 27019
```

- `rs.initiate()`
- `rs.status()`
- `rs.add()`

mongoDB

{name: "mongo", type:"DB"}

Config

- `conf = rs.conf()`
- `conf.members[2].priority = 100`
- `rs.reconfig(conf)`
- Options
  - Hidden
  - Slave delay
  - Priority
  - Arbiter

Tagging

```
{
    _id : "mySet",
    members : [
        {_id : 0, host : "A", tags : {"dc": "ny"}},
        {_id : 1, host : "B", tags : {"dc": "ny"}},
        {_id : 2, host : "C", tags : {"dc": "sf"}},
        {_id : 3, host : "D", tags : {"dc": "sf"}},
        {_id : 4, host : "E", tags : {"dc": "cloud"}}],
    settings : {
        getLastErrorModes : {
            allDCs :  {"dc" : 3},
            someDCs : {"dc" : 2}} }
}
> db.blogs.insert({...})
> db.runCommand({getLastError : 1, w : "someDCs"})
```

mongoDB
{name: "mongo", type:"DB"}

Other commands

- `rs.help()`
- `rs.status()`
- `rs.slaveOk()`
- `db.printReplicationInfo()`
- `db.printSlaveReplicationInfo()`

mongoDB
{name: "mongo", type:"DB"}

Exercises

- Set up a 3 node replica set
- Run the command to step down a primary: `rs.stepDown()` and ensure that a secondary is elected the new primary. Bring the set back up to 3 members again
- Set a priority on a node. Terminate the primary node and practice automated failover, see what happens.
- Insert some data into primary and read it from a secondary - what happens?
- Add a new node (so 4 members now) in your RS and kill 2, does a primary get elected? See what happens.

mongoDB
{name: "mongo", type:"DB"}

Sharding

- What is sharding?
- Why shard?
- Sharded architecture
- Chunks
- MongoS balancing

mongoDB
{name: "mongo", type:"DB"}

Sharding setup

- **make a directory for each shard and a config database**

```
mkdir /data/shard1 /data/shard2 /data/config
```

- **start a config server**

```
mongod --configsvr
```

- **start a mongos**

```
mongos --configdb “xxxx.local” --chunkSize 1
```

- **start 2 mongod’s**

```
mongod --dbpath /data/shard1 --port 27020

mongod --dbpath /data/shard2 --port 27021
```

mongoDB
{name: “mongo”, type:“DB”}

Connecting to a shard

- mongo
- Will prompt with mongos
- `sh.status()`
- `sh.addShard()`
- `sh.enableSharding()`
- `sh.shardCollection()`

mongoDB

{name: "mongo", type:"DB"}

Exercises

- Setup a 2 shard cluster
- Insert some test data and shard on a particular field
- Watch what happens with the balancing
- Explore the configdb

mongoDB

{name: "mongo", type:"DB"}

Backup and Restore

- Mongodump/mongorestore
- Mongoexport/mongoimport
- File system snapshot (LVM)
- File copy

mongoDB
{name: "mongo", type:"DB"}

Exercises

- Use mongodump to dump your test database from a single mongod
- Use mongorestore to restore it
- Use mongoexport to export your test database to json
- Use mongoimport to import your database
- Mongodump from a sharded set up

mongoDB
{name: "mongo", type:"DB"}

Monitoring Server

- MMS
- Munin/Nagios
- iostat -xm 2
- top
- mongostat

mongoDB
{name: "mongo", type:"DB"}

Monitoring MongoDB

- db.serverStatus()
- db.stats()
- db.test.stats()
- db.currentOp()

mongoDB
{name: "mongo", type:"DB"}

Exercise

- Create a for-loop to insert data into MongoDB

```
for(x=0;x<10000<x++) { db.test.insert({x:x}) }
```

- Watch mongostat statistics
- Find out the current operation of MongoDB
- Find out how much memory and cpu is being utilized

mongoDB
{name: "mongo", type:"DB"}

Production Notes

- 64-bit operating system
- ext4 or efs file system
- focus on RAM not cpu
- disk type (SAS 15k, SSD) for best performance
- no atime
- ulimit = 20,000
- No NUMA, no hugepages
- 16kb readahead
- RAID 10

mongoDB
{name: "mongo", type:"DB"}

Thank you

@samuel_weaver

mongoDB

{name: "mongo", type:"DB"}