**MongoDB Administration**

The administration documentation addresses the ongoing operation and maintenance of MongoDB instances and deployments.

**Administration Concepts –**

The core administration documents address strategies and practices used in the operation of MongoDB systems and deployments**.**

* **Operational Strategies** –Higher level documentation of key concepts for the operation and maintenance of MongoDB deployments.
* **MongoDB Backup Methods** – Describes approaches and considerations for backing up a MongoDB database.

Back Up with **mongodump:**

mongodump reads data from a MongoDB database and creates high fidelity BSON files which the mongorestore tool can use to populate a MongoDB database. mongodump and mongorestore are simple and efficient tools for backing up and restoring small MongoDB deployments

Run mongodump from the system command line, not the mongo shell.

Syntax for mongodump

$ mongodump [options]

To connect to a local MongoDB instance running on port 27017 and use the default settings to export the content, run mongodump without any command-line options.

$ mongodump

To specify a host and/or port of the MongoDB instance, you can either.

* Specify the hostname and port in the [--uri connection string](https://docs.mongodb.com/manual/reference/program/mongodump/#cmdoption-mongodump-uri)

$ mongodump --uri="mongodb://mongodb0.example.com:27017" [additional options]

* Specify the hostname and port in the [--host](https://docs.mongodb.com/manual/reference/program/mongodump/#cmdoption-mongodump-host)

$ mongodump --host="mongodb0.example.com:27017" [additional options]

* Specify the hostname and port in the --host and –port

mongodump --host="mongodb0.example.com" --port=27017 [additional options]

* Connect to a Replica Set

Specify the replica set name and members in the --uri connection string

$ mongodump --uri="mongodb://mongodb0.example.com:27017,mongodb1.example.com:27017,mongodb2.example.com:27017/?replicaSet=myReplicaSetName" [additional options]

Specify the replica set name and members in the –host

$ mongodump --host="myReplicaSetName/mongodb0.example.com:27017,mongodb1.example.com:27017,mongodb2.example.com" [additional options]

Restoring with **mongorestore** command:

Run mongorestore from the system command line, not the mongo shell.

Syntax :

$ mongorestore [options] [<directory>/<BSON file>]

For example, to restore from a dump directory to a local mongod instance running on  
 port 27017.

$ mongorestore dump/

You can also restore a specific collection or collections from the dump/ directory. For   
 example, the following operation restores a single collection from corresponding data   
 files in the dump/ directory:

$ mongorestore --nsInclude=test.purchaseorders dump/

**Backup and restore with mongexport/mongoimport:**

mongoexport is a command-line tool that produces a JSON or CSV export of data   
 stored in a MongoDB instance.

mongoexport must be run directly from the system command line

$ mongoexport --collection=<coll> [options]

You must specify the collection to export. If you do not specify an output file,   
 mongoexport writes to the standard output (e.g. stdout)

To connect to a local MongoDB instance running on port 27017, you do not have to   
 specify the host or port.

For example, to export the specified collection to the specified output file from a local   
  
  
 MongoDB instance running on port 27017MongoDB instance running on port 27017

$ mongoexport --collection=events --db=reporting --out=events.json

To specify a host and/or port of the MongoDB instance, you can either –uri or –host

$ mongoexport --uri="mongodb://mongodb0.example.com:27017/reporting" --  
 collection=events --out=events.json [additional options]

$ mongoexport --host="mongodb0.example.com:27017" --collection=events --  
 db=reporting --out=events.json [additional options]

**For Replica Set** :

$ mongoexport --  
 uri="mongodb://mongodb0.example.com:27017,mongodb1.example.com:27017,mong  
 odb2.example.com:27017/reporting?replicaSet=myReplicaSetName" --  
 collection=events --out=events.json [additional options]

$ mongoexport --  
 host="myReplicaSetName/mongodb0.example.com:27017,mongodb1.example.com:27  
 017,mongodb2.example.com" --collection=events --db=reporting --out=events.json  
 [additional options]

**For Sharding** :

$ mongoexport --uri="mongodb://mongos0.example.com:27017/reporting" --  
 collection=events --out=events.json [additional options]

The mongimport command :

$mongoimport --uri   
 "mongodb://[username:password@]host1[:port1][,host2[:port2],...[,hostN[:portN]]][/[d  
 atabase][?options]]"

* **Monitoring for MongoDB** – An overview of monitoring tools, diagnostic strategies, and approaches to monitoring replica sets and sharded clusters.

There are 3 ways we can monitoring mongoDB

**mongostat** will tell you how many time database operations such as insert, query, update, delete, etc. actually occur on the server. This will give a good idea on how much the load the server is handling and will indicate whether you need additional resources on the server or maybe additional servers to distribute the load.

**mongotop** tracks and reports the current read and write activity of a MongoDB instance, and reports these statistics on a per collection basis.

The serverStatus command, or **db.serverStatus**() from the shell, returns an overview of the status of the database, with details on the disk usage, memory use, connections established to the MongoDB environment, etc.

* **Run-time Database Configuration** – Outlines common MongoDB configurations and examples of best-practice configurations for common use cases.

**The command line** and **configuration file** interfaces provide MongoDB administrators with a large number of options and settings for controlling the operation of the database system

While both interfaces provide access to the same collection of options and settings, this document primarily uses the configuration file interface. If you installed MongoDB with a package manager such as yum or apt on Linux

a default configuration file (mongod.conf in /etc/mongod.conf)has been provided as part of your installation.

On Linux systems that use the systemd init system (the systemctl command)

$ sudo systemctl start mongod

On Linux systems that use the SystemV init init system (the service command):

$ sudo service mongod start

If you installed MongoDB using a TGZ or ZIP file, you will need to create your own configuration file. A basic example configuration can be found later in this document. Once you have created a configuration file, you can start a MongoDB instance with this configuration file by using either the --config or -f options to mongod:

mongod --config /etc/mongod.conf

mongod -f /etc/mongod.conf

Modify the values in the /etc/mongod.conf file on your system to control the configuration of your database instance.

**Configure the Database: in mongod.conf**

------------------------------------------------------------------------------------------------------------------

processManagement:

fork: true

net:

bindIp: localhost

port: 27017

storage:

dbPath: /var/lib/mongo

systemLog:

destination: file

path: "/var/log/mongodb/mongod.log"

logAppend: true

storage:

journal:

enabled: true

---------------------------------------------------------------------------------------------------------------

**For Security Considerations**

net:

bindIp: localhost,10.8.0.10,192.168.4.24,/tmp/mongod.sock

security:

authorization: enabled

**For Replication Configuration**

replication:

replSetName: rs0

To enable authentication for the replica set using keyfiles , add the following keyFile option.

security:

keyFile: /srv/mongodb/keyfile

Setting keyFile enables authentication and specifies a keyfile for the replica set member to use when authenticating to each other.

**For Sharding Configuration** :

sharding:

clusterRole: configsvr

net:

bindIp: 10.8.0.12

port: 27001

replication:

replSetName: csRS

sharding:

clusterRole: shardsvr

replication:

replSetName: shardA

* **Data Management** – Core documentation that addresses issues in data management, organization, maintenance, and lifecycle management.
* **Data Center Awareness** – Presents the MongoDB features that allow application developers and database administrators to configure their deployments to be more data center aware or allow operational and location-based separation.
* **Capped Collections** – Capped collections provide a special type of size-constrained collections that preserve insertion order and can support high volume inserts.

db.createCollection( "log", { capped: true, size: 100000 } )

db.createCollection("log", { capped : true, size : 5242880, max : 5000 } )

Query a Capped Collection:

To retrieve documents in reverse insertion order, issue find() along with the sort() method with the $natural parameter set to -1, as shown in the following example:

db.cappedCollection.find().sort( { $natural: -1 } )

Check if a Collection is Capped :

db.collection.isCapped()

Convert a Collection to Capped:

db.runCommand({"convertToCapped": "mycoll", size: 100000});

**Administration Tutorials**

The administration tutorials provide specific step-by-step instructions for performing common MongoDB setup, maintenance and configuration operations.

Configuration, Maintenance, and Analysis – Describes routine management operations, including configuration and performance analysis.

* **Manage mongod Processes** – Start, configure, and manage running mongod.

**Start mongod Processes**:

To start MongoDB using all defaults, issue the following command at the system shell

$ mongod

Specify a data directory and tcp port:

$ mongod –dbPath /var/lib/data --port 27017

**Start mongod as a Daemon**:

To run a mongod process as a daemon (i.e. fork), and write its output to a log file, use the --fork and --logpath options. You must create the log directory; however, mongod will create the log file if it does not exist.

The following command starts mongod as a daemon and records log output to /var/log/mongodb/mongod.log.

$ mongod --fork --logpath /var/log/mongodb/mongod.log

**Stop mongod Processes**:

In a clean shutdown a mongod completes all pending operations, flushes all data to data files, and closes all data files. Other shutdowns are unclean and can compromise the validity of the data files.

To ensure a clean shutdown, always shutdown mongod instances using one of the following methods:

**Use shutdownServer()**

Shut down the mongod from the mongo shell using the db.shutdownServer() method as follows.

use admin

db.shutdownServer()

For systems with authorization enabled, users may only issue db.shutdownServer() when authenticated to the admin database or via the localhost interface on systems without authentication enabled.

**Use –shutdown**

$mongod –shutdown

Use CTRL-c

Use Kill

**Stop a Replica Set**

If the mongod is the primary in a replica set, the shutdown process for this mongod instance has the following steps:

check how up-to-date the secondaries are.

If no secondary is within 10 seconds of the primary, mongod will return a message that it will not shut down. You can pass the shutdown command a timeoutSecs argument to wait for a secondary to catch up.

If there is a secondary within 10 seconds of the primary, the primary will step down and wait for the secondary to catch up.

After 60 seconds or once the secondary has caught up, the primary will shut down.

**Force Replica Set Shutdown**

If there is no up-to-date secondary and you want the primary to shut down, issue the shutdown command with the force argument, as in the following mongo shell operation

db.adminCommand({shutdown : 1, force : true})

The following command issues shutdown with timeoutSecs set to 5:

db.adminCommand({shutdown : 1, timeoutSecs : 5})

or

db.shutdownServer({timeoutSecs : 5})

* **Rotate Log Files** –

When used with the --logpath option or systemLog.path setting, mongod and mongos instances report a live account of all activity and operations to a log file. When reporting activity data to a log file, by default, MongoDB only rotates logs in response to the logRotate command, or when the mongod or mongos process receives a SIGUSR1 signal from the operating system.

Archive the current log files and start new ones.

To do this, the mongod or mongos instance renames the current log file by appending a UTC timestamp to the filename, in ISODate format. It then opens a new log file, closes the old log file, and sends all new log entries to the new log file.

When we started the mongod with the following command

$ mongod -v --logpath /var/log/mongodb/server1.log

You can also explicitly specify --logRotate rename.

We can list the matching files with following command

$ ls /var/log/mongodb/server1.log\*

Rotate the log file by issuing the logRotate command from the admin database in a mongo shell.

db.adminCommand( { logRotate : 1 } )

List the new log files to view the newly-created log

$ ls /var/log/mongodb/server1.log\*

Log Rotation with --logRotate reopen

Start a mongod instance, specifying the reopen --logRotate behavior.

$ mongod -v --logpath /var/log/mongodb/server1.log --logRotate reopen –logappend

List the log file with following command

$ ls /var/log/mongodb/server1.log\*

Rotate the log file by issuing the logRotate command from the admin database in a mongo shell.

db.adminCommand( { logRotate : 1 } )

* **Backup and Recovery** – Outlines procedures for data backup and restoration with mongod instances and deployments.
* **Recover Data after an Unexpected Shutdown** – Recover data from MongoDB data files that were not properly closed or have an invalid state.
* **MongoDB Scripting** – An introduction to the scripting capabilities of the mongo shell and the scripting capabilities embedded in MongoDB instances.

**Administration Reference**

* **UNIX ulimit Settings** –Describes user resources limits (i.e. ulimit) and introduces the considerations and optimal configurations for systems that run MongoDB deployments.

Most UNIX-like operating systems, including Linux and macOS, provide ways to limit and control the usage of system resources such as threads, files, and network connections on a per-process and per-user basis. These “ulimits” prevent single users from using too many system resources. Sometimes, these limits have low default values that can cause a number of issues in the course of normal MongoDB operation

ulimit – It is Unix OS command, You can use the ulimit command at the system prompt to check system limits.

$ ulimit -a

-t: cpu time (seconds) unlimited

-f: file size (blocks) unlimited

-d: data seg size (kbytes) unlimited

-s: stack size (kbytes) 8192

-c: core file size (blocks) 0

-m: resident set size (kbytes) unlimited

-u: processes 192276

-n: file descriptors 21000

-l: locked-in-memory size (kb) unlimited

-v: address space (kb) unlimited

-x: file locks unlimited

-i: pending signals 192276

-q: bytes in POSIX msg queues 819200

-e: max nice 30

-r: max rt priority 65

-N 15: unlimited

You can change ulimit settings by issuing a command in the following form:

$ ulimit -n <value>

* **System Collections** – Introduces the internal collections that MongoDB uses to track per-database metadata, including indexes, collections, and authentication credentials.
* **Database Profiler Output** – Describes the data collected by MongoDB’s operation profiler, which introspects operations and reports data for analysis on performance and behavior.

Database Profiler :

The process of fetching and logging slow queries is known as profiling. Another tool for enhancing performance in MongoDB is Query Profiler. Mongo has the facility to log slow queries and these queries can be reviewed later.

Mongo has 3 profiling levels (0,1 and 2).

Level 0 – No profiling data available

Level 1 – Slow operations are logged

Level 2 – All Operations are logged

To see the current profiling level, command db.getProfilingLevel() is executed

db.getProfilingLevel()

Output is :

> db.getProfilingLevel()

0

>

Current profiling level is 0, we are going to change it to profiling level 2.

db.setProfilingLevel(2)

{ "was" : 0, "slowms" : 100, "ok" : 1 }

To check profile status use this command :

db.getProfilingStatus();

Fetching Data from Query Profiler:

The database profiler logs information about database operations in the system.profile   
 collection. The profiling data is available for querying from this special collection named   
 system.profile.

Below are the few examples for fetching data from collection system.profile.

To return the most recent 10 log entries in the system.profile collection, run a query  
 similar to the following:

db.system.profile.find().limit(10).sort( { ts : -1 } ).pretty()

* **Server Status Output** – Provides an example and a high level overview of the output of the serverStatus.

We can check the server status with:

db.serverStatus();

* **Journaling Mechanics** – Describes the internal operation of MongoDB’s journaling facility and outlines how the journal allows MongoDB to provide provides durability and crash resiliency.
* **Exit Codes and Statuses** – Lists the unique codes returned by mongos and mongod processes upon exit.