MongoDB Data Migration, Backup and Restore

While using MongoDB to manage your network performance and configuration data you might need to move the server, back it up, restore data, etc. There are many ways you can do this, and in this page we will describe one we used to move data from MongoDB on one server to another server.

This document will not describe details about MongoDB, it assumes some basic knowledge of MongoDB, but not much; For details about MongoDB see http://docs.mongodb.org/.

What Collections to Migrate

Opmantek are using MongoDB for your data and some of the data is small'ish, while other data is very large. MongoDB does not use tables, but calls data being stored collections. Some of these collections with Opmantek products can get quite large and you may want to consider if you need to migrate them.

Access MongoDB

```
mongo
use nmis
```

You may need to authenticate to the database, the default user and password for Opmantek Applications is -u opUserRW/op42flow42

```
db.auth("opUserRW","op42flow42");
```

List the Collections

```
db.getCollectionNames()
```

The result will look something like below, the data is in JSON format.

```
> db.getCollectionNames()
[
  "command_output_log",
  "command_outputs",
  "conversations",
  "customapps",
  "endpoints",
  "flows",
  "iana",
  "reportConfig",
  "reportData",
  "sites",
  "sumCache",
  "system.indexes",
  "system.users"
]
```

Select Collections to Migrate

For our purposes, we didn't want to save the flows and conversation data, that is too large, but we did want to migrate the endpoints information (over 3 million rows) and the opConfig data, which is in command_output_log, command_outputs

Migrating Collections from MongoDB to MongoDB

Backup and restoration of MongoDB databases follows the same scenario as migrating between MongoDB instances, which is described below. The official MongoDB documentation also has a page documenting the Backup and Restore process.

Create a Working Space

Look for a place to work. MongoDB dumps data into a directory of your choice, which will require sufficient space for the database files to fit.

For this example, I made a folder on the target machine, I was migrating data from a machine called kaos to a machine called nmisdev64, so I ssh'd to nmisdev64 and created a folder called kaos, that became my working directory.

```
ssh nmisdev64
cd /data
mkdir kaos
cd kaos
```

Dumping one or more Collections

Depending on how you have installed MongoDB you should have the command /usr/local/mongodb/bin/mongodump if not find mongodump on your computer.

Then dump a collection to a BSON file. If you have a user defined on this DB you will need to also use the following in the command line, e.g. "-u opUserRW -p op42flow42" for the default credentials.

If you want to dump all collections in one go, simply omit the "-c *somecollection*" argument in the invocation below.

```
mongodump -h kaos -d nmis -c customapps -o .
```

Once you know it is working, dump some more collections; the endpoints one in testing took a little while.

```
mongodump -h kaos -d nmis -c sites -o .
mongodump -h kaos -d nmis -c reportConfig -o .
mongodump -h kaos -d nmis -c reportData -o .
mongodump -h kaos -d nmis -c sumCache -o .
mongodump -h kaos -d nmis -c command_outputs -o .
mongodump -h kaos -d nmis -c command_output_log -o .
mongodump -h kaos -d nmis -c endpoints -o .
```

Importing one or more Collections

Now you have them on disk you can import them to another Mongo instance. The example below assumes you are going to import to a local MongoDB, and that the credentials have been set.

To restore specific collections, run the following from the folder where the dumped BSON files are located:

```
mongorestore -u opUserRW -p op42flow42 -d nmis -c customapps customapps.bson
mongorestore -u opUserRW -p op42flow42 -d nmis -c sites sites.bson
mongorestore -u opUserRW -p op42flow42 -d nmis -c reportConfig reportConfig.bson
mongorestore -u opUserRW -p op42flow42 -d nmis -c reportData reportData.bson
mongorestore -u opUserRW -p op42flow42 -d nmis -c sumCache sumCache.bson
mongorestore -u opUserRW -p op42flow42 -d nmis -c command_outputs command_outputs.bson
mongorestore -u opUserRW -p op42flow42 -d nmis -c command_output_log command_output_log.bson
mongorestore -u opUserRW -p op42flow42 -d nmis -c endpoints endpoints.bson
```

To restore all collections in bulk, simply omit the "-c *somecollection dumpfile*.bson" arguments, and replace them with the directory holding all your dumpfiles:

```
/usr/local/mongodb/bin/mongorestore -u opUserRW -p op42flow42 -d nmis .
```

Verify the Collections Exist

```
[root@nmisdev64 data]# mongo
MongoDB shell version: 2.4.1
connecting to: test
> use nmis
switched to db nmis
> db.auth("opUserRW","op42flow42");
1
> db.getCollectionNames()
[
 "command_output_log",
 "command_outputs",
 "customapps",
 "endpoints",
 "reportConfig",
 "reportData",
 "sites",
 "sumCache",
 "system.indexes",
 "system.users",
 "trend"
]
> db.endpoints.count()
3203784
```

You can tell your Opmantek Apps to start using the new MongoDB instance.