Backup and Restore in MongoDB

NIIT

Objectives

By the end of this session, you should be able to:

- ❑ Know about the MongoDB Utilities
- ❑ Exporting MongoDB Data
- ❑ Importing Data into MongoDB
- ❑ Backing up an Entire Database
- ❑ Restoring a MongoDB Database

Introduction

- It is often not safe or feasible to run queries directly against production databases, so the process of duplicating datasets onto a testing environment is quite common.
- We will examine the procedures for migrating, importing or exporting for an existing MongoDB server and setting up a new database with existing data.
- We will built up some aggregations whose output was a new collection containing summary data.

The MongoDB Utilities

- The mongo shell does not include functions for exporting, importing, backup or restore.
- MongoDB has created methods for accomplishing this, so that no scripting work or complex GUIs are needed. Hence, several utility scripts are provided that can be used to get data in or out of the database in bulk.

- These utility scripts are:
  - mongoimport
  - mongoexport
  - mongodump
  - mongorestore

Exporting MongoDB Data : mongoexport

- To move data in and out of MongoDB in bulk, the most common and generally useful utility is mongoexport.
- It is useful as it is one of the primary ways to extract large amounts of data from MongoDB in a usable format (It is important to note that mongoexport must run on a single specified database and collection.)
- Code Snippet

```
mongoexport --uri=mongodb+srv://USERNAME:PASSWORD@provendocs-fawxo.
gcp.mongodb.net/sample_mflix -quiet --limit=10 --sort="{theaterId:1}"
--collection=theaters --out=output.json
```

Using mongoexport

- mongoexport syntax :

```
mongoexport --collection=theaters
```

- but this will give error as :

```
2020-03-07-T13:16:09.152+1100 error connecting to db server: no reachable
servers
```

- Since we didn’t use any database or URI, therefore: We use

```
mongoexport --uri=mongodb+srv://USERNAME:PASSWORD@myAtlasServer.gcp.
mongodb.net/sample_mflix --collection=theaters --out=output.json
```

At the end of the output, you should see the number of exported records:

```
{"_id":{"$oid":"59a47287cfa9a3a73e51ed46"},"theaterId":952,"location":
   {"address":{"street1":"4620 Garth
Rd","city":"Baytown","state":"TX","zipcode":"77521"},"geo":
   {"type":"Point","coordinates":[-94.97554,29.774206]}}}
{"_id":{"$oid":"59a47287cfa9a3a73e51ed47"},"theaterId":953,"location":
   {"address":{"street1":"10 McKenna
Rd","city":"Arden","state":"NC","zipcode":"28704"},"geo":
   {"type":"Point","coordinates":[-82.536293,35.442486]}}}
2020-08-17T11:07:24.992+1000     [########################]   sample_mflix.
theaters   1564/1564   (100.0%)
2020-08-17T11:07:24.992+1000     exported 1564 records
```

mongoexport Options

- **--quiet:** This option reduces the amount of output sent to the command line during export.
- **--type:** This will affect how the documents are printed in the console and defaults to JSON. For example, you can export the data in Comma-Separated Value (CSV) format by specifying CSV.
- **--pretty:** This outputs the documents in a nicely formatted manner.
- **--fields:** This specifies a comma-separated list of keys in your documents to be exported, similar to an export level projection.
- **--skip:** This works similar to a query level skip, skipping documents in the export.
- **--sort:** This works similar to a query level sort, sorting documents by some keys.
- **--limit:** This works similar to a query level limit, limiting the number of documents outputted.

Importing Data into MongoDB

- Use to share the data in disk with someone we use mongoimport.
- The command supports JSON, CSV and TSV formats, meaning data extracted from other applications or manually created can still be easily added to the database using mongoimport.
- **Code snippet :**

```
mongoimport  --db=imports  --collection=contacts  --file=contacts.json
```

- The mongoimport command, when using the --uri parameter, will look as follows

```
mongoimport --uri=mongodb+srv://USERNAME:PASSWORD@myAtlasServer-fawxo.
gcp.mongodb.net/imports --collection=contacts --file=contacts.json
```

Example

- Create a file called contacts.json

```
//contacts.json
{"name": "Aragorn","location": "New Zealand","job": "Park Ranger"}
{"name": "Frodo","location": "New Zealand","job": "Unemployed"}
{"name": "Ned Kelly","location": "Australia","job": "Outlaw"}
```

- Execute the following import:

```
mongoimport --uri=mongodb+srv://USERNAME:PASSWORD@myAtlasServer-fawxo.
gcp.mongodb.net/imports --collection=contacts --file=contacts.json
```

- Output :

```
2020-08-17T20:10:38.892+1000    connected to: mongodb+srv://
[**REDACTED**]@performancetuning.98afc.g
cp.mongodb.net/imports
2020-08-17T20:10:39.150+1000    3 document(s) imported successfully. 0
document(s) failed to import.
```

- To view In JSON format use –jsonArray as:

```
mongoimport --uri=mongodb+srv://USERNAME:PASSWORD@myAtlasServer-fawxo.
gcp.mongodb.net/imports --collection=contacts --file=contacts.json
--jsonArray
```

mongoimport Options

- --quiet: This reduces the amount of output messaging from the import.
- --drop: This drops the collection before beginning import.
- --jsonArray: A JSON type only, this specifies if the file is in a JSON array format.
- --type: This can be either JSON, CSV, or TSV to specify what type of file will be imported, but the default type is JSON.
- --ignoreBlanks TSV and CSV only, this will ignore empty fields in your import file.
- --headerline : TSV and CSV only, this will assume the first line of your import file is a list of field names.
- --fields: TSV and CSV only, this will specify a comma-separated list of keys in your documents for CSV and TSV formats. This is only needed if you do not have a header line.
- --stopOnError: If specified, the import will stop on the first error it encounters.

Backing up an Entire Database

- While using mongoexport we could take an entire MongoDB server and extract all the data in each database and collection. However, we do this with one collection at a time to ensure the files correctly mapped to the original database and collection.
- Along with mongoimport and mongoexport, we can export the entire contents of a database using mongodump. We provide URI (or host and port numbers), with the mongodump command .
- By combining mongodump and mongorestore, we have a reliable way of backing up, restoring, and migrating MongoDB databases across different hardware and software configurations.

Using mongodump

- mongodump command :

```
mongodump
```

- We can run mongodump without a single parameter since the only piece of information the command needs to use is the location of your MongoDB server. If no URI or host is specified, it will attempt to create a backup of a MongoDB server running on your local system.
- By default, mongodump exports everything in our MongoDB server.

mongodump Options

- --quiet: This reduces the amount of output messaging from the dump.
- --out: This allows you to specify a different location for the export to be written to disk, by default it will create a directory called "dump" in the same directory the command is run.
- --db: This allows you to specify a single database for the command to backup, by default it will back up all databases
- --collection: This allows you to specify a single collection to backup, by default it will back up all collections.
- --excludeCollection: This allows you to specify a collection to exclude from the backup.
- --query: This allows you to specify a query document which will limit the documents being backed up to only those matching the query.
- --gzip: If enabled, the output of the export will be a compressed file in .gz format instead of a directory.

Example

```
mongodump --uri=mongodb+srv://USERNAME:PASSWORD@myAtlas-fawxo.gcp.
mongodb.net/imports --out="./backups"
        2020-08-18T12:39:51.457+1000     writing imports.newData to
2020-08-18T12:39:51.457+1000     writing imports.contacts to
2020-08-18T12:39:51.457+1000     writing imports.oldData to
2020-08-18T12:39:51.697+1000     done dumping imports.newData (5
documents)
2020-08-18T12:39:52.472+1000     done dumping imports.contacts (3
documents)
2020-08-18T12:39:52.493+1000     done dumping imports.oldData (5
documents)
```

To check :

```
╭─ ~/backups
╰─ ls
       imports/

╭─ ~/backups
╰─ ls imports
       contacts.bson              contacts.metadata.json newData.bson
       newData.metadata.json  oldData.bson              oldData.metadata.json
```

Restoring a MongoDB Database

- The exports would not be beneficial in our backup strategy unless we possess a method for loading them back into a MongoDB server. The command that complements mongodump by putting our export back into the Database is mongorestore.

- The mongorestore command is ideal for restoring a dump after a disaster or for migrating an entire MongoDB instance to a new configuration

- When put in combination with other commands, mongorestore completes the import and export lifecycle.

Using mongorestore

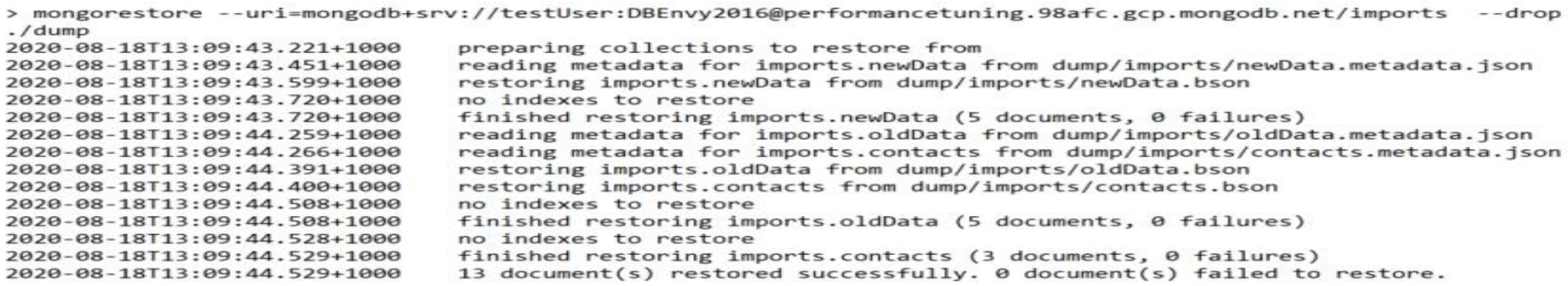
- mongorestore command:

```
mongorestore .\dump\
```

- We can specify a URI using the --uri parameter to specify the location of our MongoDB server.

```
mongodump --uri=mongodb+srv://USERNAME:PASSWORD@myAtlas-fawxo.gcp.
mongodb.net/imports --out=./dump
```

- Run mongorestore against this dump using the --drop option:



```
> mongorestore --uri=mongodb+srv://testUser:DBEnvy2016@performancetuning.98afc.gcp.mongodb.net/imports  --drop
./dump
2020-08-18T13:09:43.221+1000    preparing collections to restore from
2020-08-18T13:09:43.451+1000    reading metadata for imports.newData from dump/imports/newData.metadata.json
2020-08-18T13:09:43.599+1000    restoring imports.newData from dump/imports/newData.bson
2020-08-18T13:09:43.720+1000    no indexes to restore
2020-08-18T13:09:43.720+1000    finished restoring imports.newData (5 documents, 0 failures)
2020-08-18T13:09:44.259+1000    reading metadata for imports.oldData from dump/imports/oldData.metadata.json
2020-08-18T13:09:44.266+1000    reading metadata for imports.contacts from dump/imports/contacts.metadata.json
2020-08-18T13:09:44.391+1000    restoring imports.oldData from dump/imports/oldData.bson
2020-08-18T13:09:44.400+1000    restoring imports.contacts from dump/imports/contacts.bson
2020-08-18T13:09:44.508+1000    no indexes to restore
2020-08-18T13:09:44.508+1000    finished restoring imports.oldData (5 documents, 0 failures)
2020-08-18T13:09:44.528+1000    no indexes to restore
2020-08-18T13:09:44.529+1000    finished restoring imports.contacts (3 documents, 0 failures)
2020-08-18T13:09:44.529+1000    13 document(s) restored successfully. 0 document(s) failed to restore.
```

NIIT

The mongorestore Options

- --quiet: This reduces the amount of output messaging from the dump.
- --drop: Similar to mongoimport, the --drop option will drop the collections to be restored before restoring them, allowing you to ensure no old data remains after the command has run.
- --dryRun: This allows you to see the output of running a mongorestore without actually changing the information in the database, this is an excellent way to test your command before executing potentially dangerous operations.
- --stopOnError: If enabled, the process stops as soon as a single error occurs.
- --nsInclude: Instead of providing a database and collection specifically, this option allows you to define which namespaces (databases and collections) should be imported from the dump file.
- --nsExclude: This is the complimentary option for nsInclude, allowing you to provide a namespace pattern that is not imported when running the restore.
- --nsFrom: Using the same namespace pattern as in nsInclude and nsExclude, this parameter can be used with --nsTo to provide a mapping of namespaces in the export to new namespaces in the restored backup. This allows you to change the names of collections during your restore.

Example

- Assume we have a full mongodump created from the sample_mflix database.

```
mongorestore --uri=mongodb+srv://USERNAME:PASSWORD@myAtlasServer-fawxo.
gcp.mongodb.net --drop --nsInclude="sample_mflix.movies" dump
```

- **Output:**

```
2020-08-18T13:12:28.204+1000    [###################.....]  sample_mflix.
movies  7.53MB/9.06MB  (83.2%)
2020-08-18T13:12:31.203+1000    [#######################.]  sample_mflix.
movies  9.04MB/9.06MB  (99.7%)
2020-08-18T13:12:33.896+1000    [########################]  sample_mflix.
movies  9.06MB/9.06MB  (100.0%)
2020-08-18T13:12:33.896+1000    no indexes to restore
2020-08-18T13:12:33.902+1000    finished restoring sample_mflix.movies
(6017 documents, 0 failures)
2020-08-18T13:12:33.902+1000    6017 document(s) restored successfully. 0
document(s) failed to restore.
```

Summary

- In this session , you have learned about :
  - Four separate commands which serves as elements in a complete backup and restore lifecycle for MongoDB.
  - How to appropriately snapshot, back up, export, and restore data in case of corruption, loss, or disaster.
  - The utilities we learned are :
    - mongoexport
    - mongoimport
    - mongodump
    - mongorestore