USING MONGORESTORE FOR RESTORING MONGODB BACKUPS

It is essential to have an efficient and reliable data restoration method after backing up data during the backup and restore process. Consider the differences:

- A properly configured restoration method means users can successfully restore the data to the previous state.
- A poor restoration method makes the whole backup process ineffective, by preventing users from accessing and restoring the backed-up data.

The **mongorestore** command is the sister command of the mongodump command. You can restore the dumps (backups) created by the mongodump command into a MongoDB instance using the mongorestore command.

In this article, you will learn how to utilize the mongorestore command to restore database backups effectively.

(This article is part of our MongoDB Guide. Use the right-hand menu to navigate.)

What is mongorestore?

mongorestore is a simple utility that is used to restore backups. It can load data from either:

- A database dump file created by the mongodump command
- The standard input to a mongod or mongos instance

Starting with MongoDB 4.4, the mongorestore utility is not included in the base MongoDB server installation package. Instead, it's distributed as a separate package within the MongoDB Database

Tools package. This allows the utility to have a separate versioning scheme starting with 100.0.0.

The mongorestore utility offers support for MongoDB versions 4.4, 4.2, 4.0, and 3.6. It may also work with earlier versions of MongoDB, but compatibility is not guaranteed.

Additionally, mongorestore supports a multitude of platforms and operating systems ranging from x86 to s390x; you can see the full compatibility list in the official documentation.

Mongorestore behavior

Here is a list of things you need to know about the expected behaviors of the mongorestore utility.

- The mongorestore utility enables users to restore data to an existing database or create a new database. When restoring data into an existing database, mongorestore will only use insert commands and does not perform any kind of updates. Because of that, existing documents with a matching value for the _id field of the documents in the backup will not be overwritten by the restoration process.

This will lead to a duplicate key error during the restoration process, as shown here:

```
llection: vehicles.vehicleinformation index: _id_ dup key: { _id: ObjectId('5fbc53065f9
a624180382411') }
2021-01-09T01:00:07.033+0000    continuing through error: E11000 duplicate key error co
llection: persons.userinformation index: _id_ dup key: { _id: ObjectId('5ff8f1d37dae422
bf0a7c4f3') }
2021-01-09T01:00:07.033+0000    continuing through error: E11000 duplicate key error co
llection: persons.userinformation index: _id_ dup key: { _id: ObjectId('5ff8f1e07dae422
bf0a7c4f4') }
2021-01-09T01:00:07.034+0000    continuing through error: E11000 duplicate key error co
llection: persons.userinformation index: _id_ dup key: { _id: ObjectId('5ff8f1f27dae422
bf0a7c4f5') }
2021-01-09T01:00:07.040+0000    no indexes to restore
2021-01-09T01:00:07.040+0000    finished restoring vehicles.vehicleinformation (0 docum
ents, 5 failures)
2021-01-09T01:00:07.041+0000    no indexes to restore
2021-01-09T01:00:07.041+0000    finished restoring persons.userinformation (0 documents
, 3 failures)
2021-01-09T01:00:07.041+0000    0 document(s) restored successfully. 8 document(s) fail
ed to restore.
```

- As mongodump does not backup indexes, the mongorestore command will recreate indexes recorded by mongodump.
- The best practice when backing up and restoring a database is to use the corresponding versions of both mongodump and mongorestore. If a backup is created using a specific version of the mongodump utility, it is advisable to use its corresponding version of the mongorestore utility to perform the restore operation.
- Mongorestore will not restore the “system.profile” collection data.
- The mongorestore utility is fully compliant with FIPS (Federal Information Processing Standard) connections to perform restore operations.
- When restoring data to a MongoDB instance with access control, you need to be aware of the following scenarios:
    - If the restoring data set does not include the “system.profile“ collection and you run the mongorestore command without the --oplogReply option, the “restore” role will provide the necessary permissions to carry out the restoration process.

    - When restoring backups that include the “system.profile” collection, even though mongorestore would not restore the said collection, it will try to create a fresh “system.profile” collection. In that case, you can use the “dbAdmin” and “dbAdminAnyDatabase” roles to provide the necessary permissions.
    - To run the mongorestore command with the --oplogReply option, the user needs to create a new user-defined role with anyAction in anyResource permissions.
- You can’t use the mongorestore utility in a backup strategy when backing up and restoring sharded clusters with sharded transactions in progress. This change was introduced in MongoDB 4.2. When dealing with shared clusters, the recommended solutions would be MongoDB Cloud Manager or Ops Manager, as they can maintain the atomicity of the transactions across shards.
- The mongorestore command should be executed from the command shell of the system because it is a separate database utility.

Using MongoDB mongorestore

In this section, you will find out the basic usage of the mongorestore utility in a standalone MongoDB instance.

Basic mongorestore syntax

```
mongorestore <options> <connection-string> <directory or file to restore>
```

The basic way to restore a database is to use the mongorestore command to specify the backup directory (dump directory) without any options. This option is suitable for databases located in the localhost (127.0.0.1) using the port 27017. The restore process will create new databases and collections as needed and will also log its progress.

```
mongorestore ./dump/
```

Result:

```
barry@mongodb04:~$ mongorestore ./dump/
2021-01-08T23:02:14.794+0000    preparing collections to restore from
2021-01-08T23:02:14.796+0000    reading metadata for vehicles.vehicleinformation f
rom dump/vehicles/vehicleinformation.metadata.json
2021-01-08T23:02:14.807+0000    restoring vehicles.vehicleinformation from dump/ve
hicles/vehicleinformation.bson
2021-01-08T23:02:14.818+0000    no indexes to restore
2021-01-08T23:02:14.819+0000    finished restoring vehicles.vehicleinformation (5
documents, 0 failures)
2021-01-08T23:02:14.819+0000    5 document(s) restored successfully. 0 document(s)
 failed to restore.
barry@mongodb04:~$
```

In the above example, you can see how to successfully restore the “vehicles” database with all its collections and documents. This will create a new database named vehicles in the MongoDB instance containing all the collections and documents of the backup. You can verify the restoration process by logging into the MongoDB instance.

```
use vehicles
show collections
```

```
db.vehicleinformation.find()
```

Result:

```
> use vehicles
switched to db vehicles
> show collections
vehicleinformation
> db.vehicleinformation.find()
{ "_id" : ObjectId("5fbc52f05f9a62418038240d"), "make" : "Audi", "model" : "RS3",
"vehicle_colours" : [ "red" ], "model_year" : [ 2019, 2021 ] }
{ "_id" : ObjectId("5fbc52f55f9a62418038240e"), "make" : "BMW", "model" : "X3", "v
ehicle_colours" : [ "red", "yellow" ], "model_year" : [ 2021, 2017, 2018 ] }
{ "_id" : ObjectId("5fbc52fb5f9a62418038240f"), "make" : "Audi", "model" : "A1", "
vehicle_colours" : [ "white", "black", "dark blue", "orange" ], "model_year" : [ 2
019, 2017, 2015 ] }
{ "_id" : ObjectId("5fbc53015f9a624180382410"), "make" : "Nissan", "model" : "GTR"
 }
{ "_id" : ObjectId("5fbc53065f9a624180382411"), "make" : "Toyota", "model" : "Yari
s" }
>
```

Restoring data into a remote MongoDB instance

In order to restore data into a remote MongoDB instance, you need to establish the connection. The connection to a database can be specified using either:

- The URI connection string
- The host option
- The host and port option

Connecting using the URI option:

```
mongorestore  --uri="mongodb://<host URL/IP>:<Port>"
```

Connecting using the host option:

```
mongorestore  --host="<host URL/IP>:<Port>"
```

Connecting using host and port options:

```
mongorestore  --host="<host URL/IP>" --port=<Port>
```

The example below shows you how to restore a backup of the remote MongoDB instance. The verbose option will provide users with a detailed breakdown of the restoration process.

```
mongorestore --verbose --host="10.10.10.59" --port=27017 ./dump/
```

Result:

```
barry@mongodb04:~$ mongorestore --verbose --host="10.10.10.59" --port=27017 ./dump/
2021-01-08T23:34:45.101+0000    using write concern: &{majority false 0}
2021-01-08T23:34:45.105+0000    preparing collections to restore from
2021-01-08T23:34:45.105+0000    found collection vehicles.vehicleinformation bson to re
store to vehicles.vehicleinformation
2021-01-08T23:34:45.105+0000    found collection metadata from vehicles.vehicleinformat
ion to restore to vehicles.vehicleinformation
2021-01-08T23:34:45.106+0000    reading metadata for vehicles.vehicleinformation from d
ump/vehicles/vehicleinformation.metadata.json
2021-01-08T23:34:45.106+0000    creating collection vehicles.vehicleinformation using o
ptions from metadata
2021-01-08T23:34:45.112+0000    restoring vehicles.vehicleinformation from dump/vehicle
s/vehicleinformation.bson
2021-01-08T23:34:45.124+0000    no indexes to restore
2021-01-08T23:34:45.125+0000    finished restoring vehicles.vehicleinformation (5 docum
ents, 0 failures)
2021-01-08T23:34:45.125+0000    5 document(s) restored successfully. 0 document(s) fail
ed to restore.
barry@mongodb04:~$
```

Restoring a secure MongoDB instance

When connecting to an access-controlled MongoDB instance, you need to provide:

- Username
- Password
- Authentication database options

Additionally, mongorestore supports key-based authentications. It is necessary to ensure that the authenticated user has the required permissions/roles in carrying out the restoration process.

Authentication syntax:

```
mongorestore  --authenticationDatabase=<Database> -u=<Username> -p=<Password>
```

The following restoration command shows how to connect to a remote MongoDB server using the username and password for authentication.

```
mongorestore --host=10.10.10.59 --port=27017 --authenticationDatabase="admin"
-u="barryadmin" -p="testpassword" ./dump/
```

Result:

```
barry@mongodb04:~$ mongorestore --host=10.10.10.59 --port=27017 --authenticationDatabas
e="admin" -u="barryadmin" -p="testpassword" ./dump/
2021-01-08T23:44:04.092+0000    preparing collections to restore from
2021-01-08T23:44:04.093+0000    reading metadata for vehicles.vehicleinformation from d
ump/vehicles/vehicleinformation.metadata.json
2021-01-08T23:44:04.098+0000    restoring vehicles.vehicleinformation from dump/vehicle
s/vehicleinformation.bson
2021-01-08T23:44:04.109+0000    no indexes to restore
2021-01-08T23:44:04.109+0000    finished restoring vehicles.vehicleinformation (5 docum
ents, 0 failures)
2021-01-08T23:44:04.109+0000    5 document(s) restored successfully. 0 document(s) fail
ed to restore.
barry@mongodb04:~$
```

Selecting Databases and Collections

Using the --nsInclude option, users can specify which database or collection needs to be restored. When using the --nsInclude option, you can use the namespace pattern (Ex: "vehicles.*", "vehicles.vehicleInformation") to define which database or collection should be included.

To specify multiple namespaces, you can use the --nsInclude command multiple times in a single command. The -nsInclude command also supports wildcards to be added in the defined namespace.

The --db and --collection options are deprecated and will result in the following error.

```
2021-01-09T00:10:01.681+0000    The --db and --collection flags are deprecated for this
 use-case; please use --nsInclude instead, i.e. with --nsInclude=${DATABASE}.${COLLECTI
ON}
```

To exclude a database or a collection, you can use the --nsExclude command.

Selecting a Database/Collection:

```
mongorestore  --nsInclude=<namespace> (${DATABASE}.${COLLECTION})
```

Excluding a Database/Collection:

```
mongorestore  --nsExclude=<namespace> (${DATABASE}.${COLLECTION})
```

In the following example, you will see how to restore the complete "persons" database. You can include the whole database by specifying the "persons" namespace with the asterisk as a wild card pattern. This will restore all the data within the database.

```
mongorestore --nsInclude=persons.* --host=10.10.10.59 --port=27017 --
authenticationDatabase="admin" -u="barryadmin" -p="testpassword" ./dump/
```

Result:

```
barry@mongodb04:~$ mongorestore --nsInclude=persons.* --host=10.10.10.59 --port=27017 -
-authenticationDatabase="admin" -u="barryadmin" -p="testpassword" ./dump/
2021-01-09T00:11:08.270+0000    preparing collections to restore from
2021-01-09T00:11:08.270+0000    reading metadata for persons.userinformation from dump/
persons/userinformation.metadata.json
2021-01-09T00:11:08.277+0000    restoring persons.userinformation from dump/persons/use
rinformation.bson
2021-01-09T00:11:08.289+0000    no indexes to restore
2021-01-09T00:11:08.289+0000    finished restoring persons.userinformation (3 documents
, 0 failures)
2021-01-09T00:11:08.290+0000    3 document(s) restored successfully. 0 document(s) fail
ed to restore.
barry@mongodb04:~$
```

Restore data from an Archive File

The mongorestore utility supports restorations from an archive file. The --archive option can be used to select the archive file, and the --nsInclude and --nsExclude options can be used in conjunction with the archive option.

```
mongorestore  --archive=<file>
```

The below example illustrates how to define an archive file when restoring data. The --nsInclude option is used to specify which collection is to be restored to the database from the archive file.

```
mongorestore -v --nsInclude=vehicles.vehicleinformation --host=10.10.10.59 --port=27017 --authenticationDatabase="admin" -u="barryadmin" -p="testpassword" --archive=db.archive
```

Result:

```
barry@mongodb04:~$ mongorestore -v --nsInclude=vehicles.vehicleinformation --host=10.10.10.59 --port=27017 --authenticationDatabase="admin" -u="barryadmin" -p="testpassword" --archive=db.archive
2021-01-09T00:42:40.812+0000    using write concern: &{majority false 0}
2021-01-09T00:42:40.850+0000    archive prelude vehicles.vehicleinformation
2021-01-09T00:42:40.851+0000    preparing collections to restore from
2021-01-09T00:42:40.851+0000    found collection vehicles.vehicleinformation bson to restore to vehicles.vehicleinformation
2021-01-09T00:42:40.851+0000    found collection metadata from vehicles.vehicleinformation to restore to vehicles.vehicleinformation
2021-01-09T00:42:40.865+0000    reading metadata for vehicles.vehicleinformation from archive 'db.archive'
2021-01-09T00:42:40.868+0000    creating collection vehicles.vehicleinformation using options from metadata
2021-01-09T00:42:40.872+0000    restoring vehicles.vehicleinformation from archive 'db.archive'
2021-01-09T00:42:40.884+0000    no indexes to restore
2021-01-09T00:42:40.884+0000    finished restoring vehicles.vehicleinformation (5 documents, 0 failures)
2021-01-09T00:42:40.884+0000    5 document(s) restored successfully. 0 document(s) failed to restore.
barry@mongodb04:~$
```

Restoring data from a Compressed File

The mongodump utility uses the --gzip option to compress the individual JSON and BSON files. These compressed backups can also be used to restore the database. The compressed file can also be filtered using the --nsInclude and --nsExclude options.

```
mongorestore --gzip
```

You can restore a compressed MongoDB backup using the following commands. The compressed backup is stored in the “backupzip” directory.

```
mongorestore --gzip -v --nsInclude=vehicles.vehicleinformation --host=10.10.10.59 --port=27017 --authenticationDatabase="admin" -u="barryadmin" -p="testpassword" ./backupzip/
```

Result:

```
barry@mongodb04:~$ ls backupzip/
vehicles
barry@mongodb04:~$ ls backupzip/vehicles/
vehicleinformation.bson.gz  vehicleinformation.metadata.json.gz
barry@mongodb04:~$
barry@mongodb04:~$ mongorestore --gzip -v --nsInclude=vehicles.vehicleinformation --hos
t=10.10.10.59 --port=27017 --authenticationDatabase="admin" -u="barryadmin" -p="testpas
sword" ./backupzip/
2021-01-09T00:55:05.031+0000    using write concern: &{majority false 0}
2021-01-09T00:55:05.057+0000    preparing collections to restore from
2021-01-09T00:55:05.057+0000    found collection vehicles.vehicleinformation bson to re
store to vehicles.vehicleinformation
2021-01-09T00:55:05.057+0000    found collection metadata from vehicles.vehicleinformat
ion to restore to vehicles.vehicleinformation
2021-01-09T00:55:05.057+0000    reading metadata for vehicles.vehicleinformation from b
ackupzip/vehicles/vehicleinformation.metadata.json.gz
2021-01-09T00:55:05.058+0000    creating collection vehicles.vehicleinformation using o
ptions from metadata
2021-01-09T00:55:05.063+0000    restoring vehicles.vehicleinformation from backupzip/ve
hicles/vehicleinformation.bson.gz
2021-01-09T00:55:05.075+0000    no indexes to restore
2021-01-09T00:55:05.075+0000    finished restoring vehicles.vehicleinformation (5 docum
ents, 0 failures)
2021-01-09T00:55:05.075+0000    5 document(s) restored successfully. 0 document(s) fail
ed to restore.
barry@mongodb04:~$
```

The same process can be applied to a compressed archive file. The below example shows how to restore data from a compressed archive file.

```
mongorestore --gzip -v --nsInclude=vehicles.vehicleinformation --
host=10.10.10.59 --port=27017 --authenticationDatabase="admin" -
u="barryadmin" -p="testpassword" --archive=db.archive
```

Result:

```
barry@mongodb04:~$ mongorestore --gzip -v --nsInclude=vehicles.vehicleinformation --hos
t=10.10.10.59 --port=27017 --authenticationDatabase="admin" -u="barryadmin" -p="testpas
sword" --archive=db.archive
2021-01-09T00:58:26.791+0000    using write concern: &{majority false 0}
2021-01-09T00:58:26.833+0000    archive prelude vehicles.vehicleinformation
2021-01-09T00:58:26.835+0000    preparing collections to restore from
2021-01-09T00:58:26.837+0000    found collection vehicles.vehicleinformation bson to re
store to vehicles.vehicleinformation
2021-01-09T00:58:26.837+0000    found collection metadata from vehicles.vehicleinformat
ion to restore to vehicles.vehicleinformation
2021-01-09T00:58:26.847+0000    reading metadata for vehicles.vehicleinformation from a
rchive 'db.archive'
2021-01-09T00:58:26.848+0000    creating collection vehicles.vehicleinformation using o
ptions from metadata
2021-01-09T00:58:26.852+0000    restoring vehicles.vehicleinformation from archive 'db.
archive'
2021-01-09T00:58:26.864+0000    no indexes to restore
2021-01-09T00:58:26.864+0000    finished restoring vehicles.vehicleinformation (5 docum
ents, 0 failures)
2021-01-09T00:58:26.864+0000    5 document(s) restored successfully. 0 document(s) fail
ed to restore.
barry@mongodb04:~$
```

Restoring data from Standard Input

The mongorestore command enables users to read data from standard inputs and use that data in the restoration process. You can read the data by providing the --archive option without the filename.

```
mongodump  --archive | mongorestore  –archive
```

The following example shows how to create a backup from a secure MongoDB database using mongodump and pass it as standard input to the mongorestore command to be restored in an insecure remote MongoDB instance.

```
mongodump --host=10.10.10.59 --port=27017 --authenticationDatabase="admin" -u="barryadmin" -p="testpassword" --db=vehicles --archive | mongorestore --host=10.10.10.58 --port=27018 --archive
```

Result:

```
barry@mongodb04:~$ mongodump --host=10.10.10.59 --port=27017 --authenticationDatabase="admin" -u="barryadmin" -p="testpassword" --db=vehicles --archive | mongorestore --host=10.10.10.58 --port=27018 --archive
2021-01-09T02:08:03.925+0000    writing vehicles.vehicleinformation to archive on stdout
2021-01-09T02:08:03.935+0000    done dumping vehicles.vehicleinformation (5 documents)
2021-01-09T02:08:03.937+0000    preparing collections to restore from
2021-01-09T02:08:03.960+0000    reading metadata for vehicles.vehicleinformation from archive on stdin
2021-01-09T02:08:03.964+0000    restoring vehicles.vehicleinformation from archive on stdin
2021-01-09T02:08:03.976+0000    no indexes to restore
2021-01-09T02:08:03.977+0000    finished restoring vehicles.vehicleinformation (5 documents, 0 failures)
2021-01-09T02:08:03.977+0000    5 document(s) restored successfully. 0 document(s) failed to restore.
barry@mongodb04:~$
```

You can verify the restoration process by checking the databases with the remote server. This can be done by executing a JavaScript code using --eval tag. Using the “listDatabases” admin command, you will be able to list all the databases within the remote MongoDB instance.

```
mongo --host=10.10.10.58 --port=27018 --quiet --eval 'printjson(db.adminCommand( { listDatabases: 1 } ))'
```

Result:

```
barry@mongodb04:~$ mongo --host=10.10.10.58 --port=27018 --quiet --eval 'printjson(db.a
dminCommand( { listDatabases: 1 } ))'
{
        "databases" : [
                {
                        "name" : "admin",
                        "sizeOnDisk" : 65536,
                        "empty" : false
                },
                {
                        "name" : "cars",
                        "sizeOnDisk" : 425984,
                        "empty" : false
                },
                {
                        "name" : "config",
                        "sizeOnDisk" : 593920,
                        "empty" : false
                },
                {
                        "name" : "food",
                        "sizeOnDisk" : 98304,
                        "empty" : false
                },
                {
                        "name" : "local",
                        "sizeOnDisk" : 872448,
                        "empty" : false
                },
                {
                        "name" : "persons",
                        "sizeOnDisk" : 131072,
                        "empty" : false
                },
                {
                        "name" : "school",
                        "sizeOnDisk" : 131072,
                        "empty" : false
                },
                {
                        "name" : "vehicles",
                        "sizeOnDisk" : 40960,
                        "empty" : false
                }
        ],
```

mongorestore & mongodump

This tutorial offers in-depth knowledge about the MongoDB mongorestore utility and how it can be used to restore backups created by mongodump. The mongorestore offers a convenient and efficient way of restoring database backups.

The combination of mongodump with mongorestore provides small scale database administrators with a complete backup strategy.

Related reading

- BMC Machine Learning & Big Data Blog
- How To Use mongodump for MongoDB Backups
- Top MongoDB Commands You Need to Know
- Snowflake Guide

- Data Storage Explained: Data Lake vs Warehouse vs Database
- What Is DBaaS? Database-as-a-Service Explained