HOW TO USE MONGODUMP FOR MONGODB BACKUPS

Maintaining backups is vital for every organization. Data backups act as a safety measure where data can be recovered or restored in case of an emergency. Typically, you create database backups by replicating the database, using either:

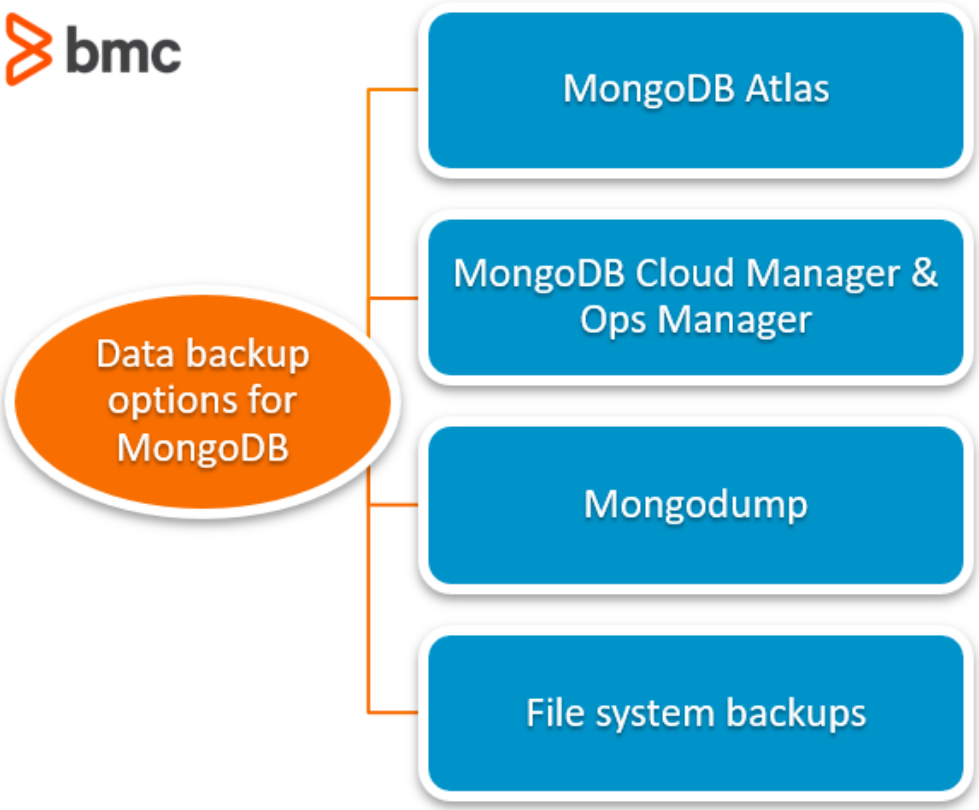
- Built-in tools
- Specialized external backup services

MongoDB offers multiple inbuilt backup options depending on the MongoDB deployment method you use. We'll look briefly at the options, but then we'll show you how to utilize one particular option—MongoDB mongodump—for the backup process.

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

Built-in backups in MongoDB

Here are the several options you have for backing up your data in MongoDB:



MongoDB Atlas Backups

If MongoDB is hosted in MongoDB Atlas cloud database service, the Altas service provides automated continuous incremental backups.

Additionally, Altas can be used to create cloud provider snapshots, where local database snapshots are created using the underlying cloud providers' snapshot functionality.

MongoDB Cloud Manager or Ops Manager

Cloud Manager is a hosted backup, monitoring, and automation service for MongoDB. Cloud Manager enables easy backup and restores functionality while providing offsite backups.

Ops Manager provides the same functionality as Cloud Manager but it can be deployed as an on-premise solution.

MongoDB mongodump

Mongodump is a simple MongoDB backup utility that creates high fidelity BSON files from an underlying database. These files can be restored using the **mongorestore** utility.

Mongodump is an ideal backup solution for small MongoDB instances due to its ease of use and portability.

File system backups

In this method, you merely keep copies of the underlying data files of a MongoDB installation. We can utilize snapshots if the file system supports it.

Another way is to use a tool like rsync where we can directly copy the data files to a backup directory.

What is MongoDB mongodump?

The **mongodump** is a utility for creating database backups. The utility can be used for creating binary export of the database contents. Mongodump can export data from both mongod and mongos instances, allowing you to create backups from:

- A standalone, replica set
- A sharded cluster of MongoDB deployments

Before MongoDB 4.4, mongodump was released alongside the MongoDB server and used matched versioning. The new iterations of mongodump are released as a separate utility in MongoDB Database Tools. Mongodump guarantees compatibility with MongoDB 4.4, 4.2, 4.0, and 3.6.

The mongodump utility is supported on most x86_64 platforms and some of ARM64, PPC64LE, and s390x platforms. You can find the full list of platforms that mongodump is compatible with from their documentation.

mongodump actions

The following list breaks down the expected behaviors and limitations of the mongodump utility.

- The mongodump utility directs its read operations to the primary member of a replica set, making the default read preference to primary.
- The backup operation will exclude the "local" database and only captures the documents excluding the index data. These indexes must be rebuilt after a restoration process.
- When it comes to backing up read-only views, mongodump only captures metadata of views. If you want to capture documents within a view, use the "--viewsAsCollections" flag.
- To ensure maximum compatibility, use Extended JSON v2.0 (Canonical) for mongodump metadata files. It is recommended to use the corresponding versions of mongodump and mongorestore in backup and restore operations.
- The mongodump command will overwrite the existing files within the given backup folder. The default location for backups is the dump/ folder.
- When the WiredTiger storage engine is used in a MongoDB instance, the output will be uncompressed data.
- Backup operations using mongodump is dependent on the available system memory. If the data set is larger than the system memory, the mongodump utility will push the working set out of memory.
- If access control is configured to access the MongoDB database, users must have enough privileges to each database to make backups. MongoDB has a built-in backup role with required privileges to backup any database.
- MongoDB allows mongodump to be a part of the backup strategy for standalone or a replica set.
- Starting with MongoDB 4.2, mongodump cannot be used as a part of the backup strategy when backing up sharded clusters that have sharded transactions in progress. In these instances, it is recommended to use a solution like MongoDB Cloud Manager or Ops Manager, which maintain the atomicity in transactions across shards.

- The mongodump command must be executed from the system command shell as it is a separate utility.
- There is no option for incremental backups. All backups will make a full copy of the database.

MongoDB Database Tools

MongoDB Database Tools are a collection of command-line utilities that help with the maintenance and administration of a MongoDB instance. The MongoDB Database tools are compatible in these environments:

- Windows
- Linux
- macOS

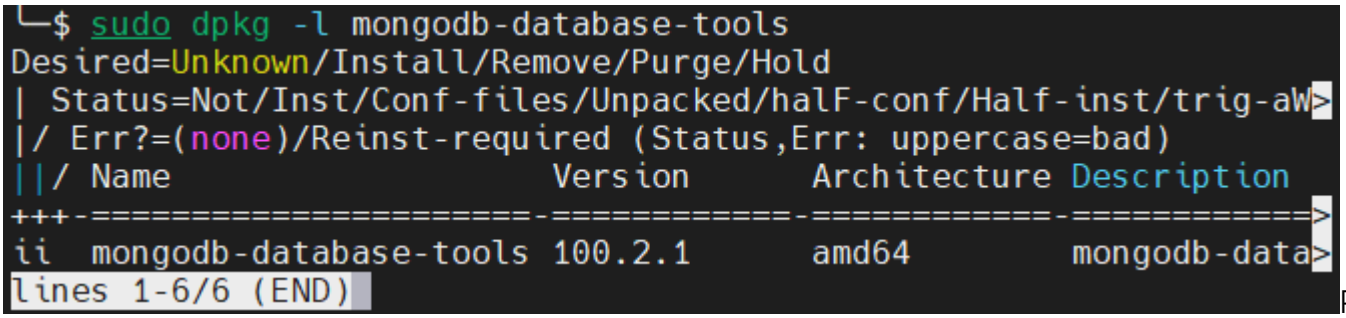
In this section, we will take a look at how we can install the Database Tools on a Linux server.

Checking for Database Tools

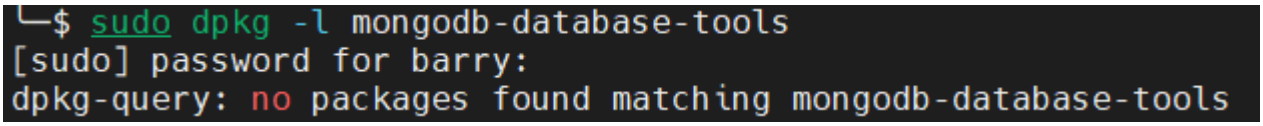
To check if the database tools are already installed on the system, we can use the following command.

```
sudo dpkg -l mongodb-database-tools
```

Result for Database Tools installed:



Result for Database Tools unavailable:



Installing Database Tools

If your system doesn't have Database Tools, here's how to install it.

The MongoDB download center provides the latest version of MongoDB Database Tools. Download the latest version according to your platform and package type. In a CLI environment, we can copy the download link and use **wget** or **curl** to download the package.

In the example below, we will be using the Database Tools version 100.2.1 for Ubuntu as a deb package and then install using the downloaded file.

```
curl -o mongodb-database-tools-ubuntu2004-x86_64-100.2.1.deb
https://fastdl.mongodb.org/tools/db/mongodb-database-tools-ubuntu2004-x86_64-
```

```
100.2.1.deb
sudo apt install ./mongodb-database-tools-ubuntu2004-x86_64-100.2.1.deb
```

Result:

```
└─$ curl -o mongodb-database-tools-ubuntu2004-x86_64-100.2.1.deb h
ttps://fastdl.mongodb.org/tools/db/mongodb-database-tools-ubuntu20
04-x86_64-100.2.1.deb
  % Total    % Received % Xferd  Average Speed   Time    Time
Time  Current
                                 Dload  Upload   Total   Spent
Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --
  0 51.9M    0  101k    0     0  64362      0  0:14:05  0:00:01  0
 12 51.9M   12 6668k    0     0  2548k      0  0:00:20  0:00:02  0
 37 51.9M   37 19.7M    0     0  5542k      0  0:00:09  0:00:03  0
 58 51.9M   58 30.2M    0     0  6706k      0  0:00:07  0:00:04  0
 78 51.9M   78 40.9M    0     0  7441k      0  0:00:07  0:00:05  0
 99 51.9M   99 51.8M    0     0  8028k      0  0:00:06  0:00:06 --
100 51.9M  100 51.9M    0     0  8028k      0  0:00:06  0:00:06 --
:--:-- 11.3M
┌─barry@ubser001 ~
└─$ sudo apt install ./mongodb-database-tools-ubuntu2004-x86_64-10
0.2.1.deb
Reading package lists... Done
Building dependency tree
Reading state information... Done
```

Using MongoDB mongodump

In this section, we will cover the basic usage of mongodump utility in a standalone MongoDB instance.

Basic mongodump Syntax

```
mongodump <options> <connection-string>
```

The most basic method to create a backup is to use the mongodump command without any options. This will assume the database is located in localhost (127.0.0.1) and using port 27017 with no authentication requirements. The backup process will create a dump folder in the current directory.

```
mongodump
```

Result:

```
barry@mongodb04:~$ mongodump
2021-01-02T19:45:35.488+0000    writing admin.system.users to dump/admin/system.us
ers.bson
2021-01-02T19:45:35.490+0000    done dumping admin.system.users (3 documents)
2021-01-02T19:45:35.490+0000    writing admin.system.version to dump/admin/system.
version.bson
2021-01-02T19:45:35.492+0000    done dumping admin.system.version (2 documents)
2021-01-02T19:45:35.493+0000    writing vehicles.vehicleinformation to dump/vehicl
es/vehicleinformation.bson
2021-01-02T19:45:35.494+0000    done dumping vehicles.vehicleinformation (5 docume
nts)
```

We can navigate to the dump folder to verify the created backups.

Backing up a remote MongoDB instance

We can specify a host and a port using the --uri connection string.

Connect using the uri option:

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

Connect using the host option:

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

Connect using host and port options:

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

The following example demonstrates how to create a backup of the remote MongoDB instance:

```
mongodump --host="10.10.10.59" --port=27017
```

Result:

```
barry@mongodb04:~$ mongodump --host="10.10.10.59" --port=27017
2021-01-02T20:18:19.702+0000    writing admin.system.users to dump/admin/system.us
ers.bson
2021-01-02T20:18:19.703+0000    done dumping admin.system.users (3 documents)
2021-01-02T20:18:19.703+0000    writing admin.system.version to dump/admin/system.
version.bson
2021-01-02T20:18:19.703+0000    done dumping admin.system.version (2 documents)
2021-01-02T20:18:19.704+0000    writing vehicles.vehicleinformation to dump/vehicl
es/vehicleinformation.bson
2021-01-02T20:18:19.704+0000    done dumping vehicles.vehicleinformation (5 docume
nts)
```

Backing up a secure MongoDB instance

If we want to connect to a MongoDB instance with access-control, we need to provide:

- Username
- Password
- Authentication database options

Authentication Syntax

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

Selecting a collection:

```
mongodump  --db=<Backup Target - Database> --collection=<Collection Name>
```

Excluding a collection:

```
mongodump  --db=<Backup Target - Database> --excludeCollection=<Collection Name>
```

In the following example, we define the "vehicleinformation" collection as the only backup target.

```
mongodump --host=10.10.10.59 --port=27017 --authenticationDatabase="admin" -u="barryadmin" -p="testpassword" --db=vehicles --collection=vehicleinformation
```

Result:

```
barry@mongodb04:~$ mongodump --host=10.10.10.59 --port=27017 --authenticationDatabase="admin" -u="barryadmin" -p="testpassword" --db=vehicles --collection=vehicleinformation
2021-01-02T21:46:53.735+0000    writing vehicles.vehicleinformation to dump/vehicles/vehicleinformation.bson
2021-01-02T21:46:53.736+0000    done dumping vehicles.vehicleinformation (5 documents)
```

Changing the backup directory

The --out option can be used to specify the location of the backup folder.

```
mongodump --out=<Directory Location>
```

Let us change the backup directory to the "dbbackup" folder.

```
mongodump --host=10.10.10.59 --port=27017 --authenticationDatabase="admin" -u="barryadmin" -p="testpassword" --out=dbbackup
```

Result:

```
barry@mongodb04:~$ mongodump --host=10.10.10.59 --port=27017 --authenticationDatabase="admin" -u="barryadmin" -p="testpassword" --out=dbbackup
2021-01-02T21:47:57.371+0000    writing admin.system.users to dbbackup/admin/system.users.bson
2021-01-02T21:47:57.372+0000    done dumping admin.system.users (3 documents)
2021-01-02T21:47:57.372+0000    writing admin.system.version to dbbackup/admin/system.version.bson
2021-01-02T21:47:57.373+0000    done dumping admin.system.version (2 documents)
2021-01-02T21:47:57.373+0000    writing vehicles.vehicleinformation to dbbackup/vehicles/vehicleinformation.bson
2021-01-02T21:47:57.374+0000    done dumping vehicles.vehicleinformation (5 documents)
```

Creating an archive file

The mongodump utility allows us to create an archive file. The --archive option can be used to specify the file. If no file is specified the output will be written to standard output (**stdout**).

The --archive option cannot be used in conjunction with the --out option.

```
mongodump --archive=<file>
```

The below example demonstrates how we can define an archive file.

```
mongodump --host=10.10.10.59 --port=27017 --authenticationDatabase="admin" -u="barryadmin" -p="testpassword" --archive=db.archive
```

Result:

```
barry@mongodb04:~$ mongodump --host=10.10.10.59 --port=27017 --authenticationDatabase="admin" -u="barryadmin" -p="testpassword" --archive=db.archive
2021-01-02T21:43:20.547+0000    writing admin.system.users to archive 'db.archive'
2021-01-02T21:43:20.550+0000    done dumping admin.system.users (3 documents)
2021-01-02T21:43:20.550+0000    writing admin.system.version to archive 'db.archive'
2021-01-02T21:43:20.552+0000    done dumping admin.system.version (2 documents)
2021-01-02T21:43:20.553+0000    writing vehicles.vehicleinformation to archive 'db.archive'
2021-01-02T21:43:20.567+0000    done dumping vehicles.vehicleinformation (5 documents)
```

Compressing the backup

The backup files can be compressed using the --gzip option. This option will compress the individual JSON and BSON files.

```
mongodump --gzip
```

Let's compress the complete MongoDB database.

```
mongodump --host=10.10.10.59 --port=27017 --authenticationDatabase="admin" -u="barryadmin" -p="testpassword" --gzip
```

Result:

```
barry@mongodb04:~$ mongodump --host=10.10.10.59 --port=27017 --authenticationDatab
ase="admin" -u="barryadmin" -p="testpassword" --gzip
2021-01-02T21:49:08.499+0000    writing admin.system.users to dump/admin/system.us
ers.bson.gz
2021-01-02T21:49:08.500+0000    done dumping admin.system.users (3 documents)
2021-01-02T21:49:08.500+0000    writing admin.system.version to dump/admin/system.
version.bson.gz
2021-01-02T21:49:08.501+0000    done dumping admin.system.version (2 documents)
2021-01-02T21:49:08.501+0000    writing vehicles.vehicleinformation to dump/vehicl
es/vehicleinformation.bson.gz
2021-01-02T21:49:08.503+0000    done dumping vehicles.vehicleinformation (5 docume
nts)
barry@mongodb04:~$ ls dump/
admin  vehicles
barry@mongodb04:~$ ls dump/vehicles/
vehicleinformation.bson.gz  vehicleinformation.metadata.json.gz
```

In this article, we learned about MongoDB mongodump and how mongodump can be used to create backups and manage database backups.

Related reading

- BMC Machine Learning & Big Data Blog
- Top MongoDB Commands You Need to Know, part of our MongoDB Guide
- Snowflake Guide
- Data Storage Explained: Data Lake vs Warehouse vs Database
- What Is DBaaS? Database-as-a-Service Explained