**How To Install MongoDB on CentOS 7**

**Introduction**

MongoDB is a document-oriented database that is free and open-source. It is classified as a NoSQL database because it does not rely on a traditional table-based relational database structure. Instead, it uses JSON-like documents with dynamic schemas. Unlike relational databases, MongoDB does not require a predefined schema before you add data to a database. You can alter the schema at any time and as often as is necessary without having to setup a new database with an updated schema.

This tutorial guides you through installing MongoDB Community Edition on a CentOS 7 server.

**Prerequisites**

Before following this tutorial, make sure you have a regular, non-root user with sudo privileges. You can learn more about how to set up a user with these privileges from our guide, [How To Create a Sudo User on CentOS](https://www.digitalocean.com/community/tutorials/how-to-create-a-sudo-user-on-centos-quickstart).

**Step 1 – Adding the MongoDB Repository**

The mongodb-org package does not exist within the default repositories for CentOS. However, MongoDB maintains a dedicated repository. Let’s add it to our server.

With the vi editor, create a .repo file for yum, the package management utility for CentOS:

* sudo vi /etc/yum.repos.d/mongodb-org.repo

Copy

Then, visit the [Install on Red Hat](https://docs.mongodb.com/manual/tutorial/install-mongodb-on-red-hat/#configure-the-package-management-system-yum) section of MongoDB’s documentation and add the repository information for the latest stable release to the file:

/etc/yum.repos.d/mongodb-org.repo

[mongodb-org-3.4]

name=MongoDB Repository

baseurl=https://repo.mongodb.org/yum/redhat/$releasever/mongodb-org/3.4/x86\_64/

gpgcheck=1

enabled=1

gpgkey=https://www.mongodb.org/static/pgp/server-3.4.asc

Copy

Save and close the file.

Before we move on, we should verify that the MongoDB repository exists within the yum utility. The repolist command displays a list of enabled repositories:

* yum repolist

Copy

Output

. . .

repo id repo name

base/7/x86\_64 CentOS-7 - Base

extras/7/x86\_64 CentOS-7 - Extras

mongodb-org-3.2/7/x86\_64 MongoDB Repository

updates/7/x86\_64 CentOS-7 - Updates

. . .

With the MongoDB Repository in place, let’s proceed with the installation.

**Step 2 – Installing MongoDB**

We can install the mongodb-org package from the third-party repository using the yum utility.

* sudo yum install mongodb-org

Copy

There are two Is this ok [y/N]: prompts. The first one permits the installation of the MongoDB packages and the second one imports a GPG key. The publisher of MongoDB signs their software and yum uses a key to confirm the integrity of the downloaded packages. At each prompt, type Y and then press the ENTER key.

Next, start the MongoDB service with the systemctl utility:

* sudo systemctl start mongod

Copy

Although we will not use them in this tutorial, you can also change the state of the MongoDB service with the reload and stop commands.

The reload command requests that the mongod process reads the configuration file, /etc/mongod.conf, and applies any changes without requiring a restart.

* sudo systemctl reload mongod

Copy

The stop command halts all running mongod processes.

* sudo systemctl stop mongod

Copy

The systemctl utility did not provide a result after executing the start command, but we can check that the service started by viewing the end of the mongod.log file with the tail command:

* sudo tail /var/log/mongodb/mongod.log

Copy

Output

. . .

[initandlisten] waiting for connections on port 27017

An output of **waiting for a connection** confirms that MongoDB has started successfully and we can access the database server with the MongoDB Shell:

* mongo

Copy

**Note:** When you launched the MongoDB Shell you may have seen a warning like this:

\*\* WARNING: soft rlimits too low. rlimits set to 4096 processes, 64000 files. Number of processes should be at least 32000 : 0.5 times number of files.

MongoDB is a threaded application. It can launch additional processes to handle its workload. The warning states that for MongoDB to be most effective the number of processes that it is authorized to spin up should be half that of the number of files that it can have open at any given time. To resolve the warning, alter the processes soft rlimit value for mongod by editing the 20-nproc.conf file:

* sudo vi /etc/security/limits.d/20-nproc.conf

Copy

Add the following line to the end of file:

/etc/security/limits.d/20-nproc.conf

. . .

mongod soft nproc 32000

Copy

For the new limit to be available to MongoDB, restart it using the systemctl utility:

* sudo systemctl restart mongod

Copy

Afterward, when you connect to the MongoDB Shell, the warning should cease to exist.

To learn how to interact with MongoDB from the shell, you can review the output of the db.help() method which provides a list of methods for the **db** object.

* db.help()

Copy

Output

DB methods:

db.adminCommand(nameOrDocument) - switches to 'admin' db, and runs command [ just calls db.runCommand(...) ]

db.auth(username, password)

db.cloneDatabase(fromhost)

db.commandHelp(name) returns the help for the command

db.copyDatabase(fromdb, todb, fromhost)

db.createCollection(name, { size : ..., capped : ..., max : ... } )

db.createUser(userDocument)

db.currentOp() displays currently executing operations in the db

db.dropDatabase()

. . .

Leave the mongod process running in the background, but quit the shell with the exit command:

* exit

Copy

Output

Bye

**Step 3 – Verifying Startup**

Because a database-driven application cannot function without a database, we’ll make sure that the MongoDB daemon, mongod, will start with the system.

Use the systemctl utility to check its startup status:

* systemctl is-enabled mongod; echo $?

Copy

An output of zero confirms an enabled daemon, which we want. A one, however, confirms a disabled daemon that will not start.

Output

. . .

enabled

0

In the event of a disabled daemon, use the systemctl utility to enable it:

* sudo systemctl enable mongod

Copy

We now have a running instance of MongoDB that will automatically start following a system reboot.

**Step 4 – Importing an Example Dataset (Optional)**

Unlike other database servers, MongoDB does not come with data in its test database. Since we don’t want to experiment with new software using production data, we will download a sample dataset from the “[Import Example Dataset](https://docs.mongodb.com/getting-started/shell/import-data/)” section of the “Getting Started with MongoDB” documentation. The JSON document contains a collection of restaurants, which we’ll use to practice interacting with MongoDB and avoid causing harm to sensitive data.

Start by moving into a writable directory:

* cd /tmp

Copy

Use the curl command and the link from MongoDB to download the JSON file:

* curl -LO https://raw.githubusercontent.com/mongodb/docs-assets/primer-dataset/primer-dataset.json

Copy

The mongoimport command will insert the data into the **test** database. The --db flag defines which database to use while the --collection flag specifies where in the database the information will be stored, and the --file flag tells the command which file to perform the import action on:

* mongoimport --db test --collection restaurants --file /tmp/primer-dataset.json

Copy

The output confirms the importing of the data from the primer-dataset.json file:

Output

connected to: localhost

imported 25359 documents

With the sample dataset in place, we’ll perform a query against it.

Relaunch the MongoDB Shell:

* mongo

Copy

The shell selects the test database by default, which is where we imported our data.

Query the **restaurants** collection with the find() method to display a list of all the restuarants in the dataset. Since the collection contains over 25,000 entries, use the optional limit() method to reduce the output of the query to a specified number. Additionally, the pretty() method makes the information more human-readable with newlines and indentations.

* db.restaurants.find().limit( 1 ).pretty()

Copy

Output

{

"\_id" : ObjectId("57e0443b46af7966d1c8fa68"),

"address" : {

"building" : "1007",

"coord" : [

-73.856077,

40.848447

],

"street" : "Morris Park Ave",

"zipcode" : "10462"

},

"borough" : "Bronx",

"cuisine" : "Bakery",

"grades" : [

{

"date" : ISODate("2014-03-03T00:00:00Z"),

"grade" : "A",

"score" : 2

},

{

"date" : ISODate("2013-09-11T00:00:00Z"),

"grade" : "A",

"score" : 6

},

{

"date" : ISODate("2013-01-24T00:00:00Z"),

"grade" : "A",

"score" : 10

},

{

"date" : ISODate("2011-11-23T00:00:00Z"),

"grade" : "A",

"score" : 9

},

{

"date" : ISODate("2011-03-10T00:00:00Z"),

"grade" : "B",

"score" : 14

}

],

"name" : "Morris Park Bake Shop",

"restaurant\_id" : "30075445"

}

You can continue using the sample dataset to familiarize yourself with MongoDB or delete it with the db.restaurants.drop() method:

* db.restaurants.drop()

Copy

Lastly, quit the shell with the exit command:

* exit

Copy

Output

Bye