How to Install and Secure MongoDB on Ubuntu 16.04

Posted February 24, 2017 102.9k MONGODB SECURITY

By: Melissa Anderson

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.

In Part One of this tutorial we'll use the MongoDB Repository to install the latest version of MongoDB. In Part Two, we'll enable authentication to secure it on the local system. Finally, in Part Three, we'll show how to more securely allow remote connections if they're needed.

Prerequisites

To follow this tutorial, you will need:

- **One Ubuntu 16.04 server** configured with a non-root `sudo` user and a firewall by following the Ubuntu 16.04 initial server setup guide.

When this is in place, you're ready to follow along.

Part One: Setting Up the Server

Step 1 — Adding the MongoDB Repository

MongoDB is already included in Ubuntu package repositories, but the official MongoDB repository provides the most up-to-date version and is the recommended way of installing the software. In this step, we will add this official repository to our server.

Ubuntu ensures the authenticity of software packages by verifying that they are signed with GPG keys, so we first have to import the key for the official MongoDB repository.

```
$ sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv 0C49F3730359A14518585931BC
```

The following output confirms that we've successfully imported the key:

Output

```
Executing: /tmp/tmp.IdwenTia0s/gpg.1.sh --keyserver
hkp://keyserver.ubuntu.com:80
--recv
0C49F3730359A14518585931BC711F9BA15703C6
gpg: requesting key A15703C6 from hkp server keyserver.ubuntu.com
gpg: key A15703C6: public key "MongoDB 3.4 Release Signing Key <packaging@mongodb.com>" import
gpg: Total number processed: 1
gpg:               imported: 1  (RSA: 1)
```

Next, we'll add MongoDB repository details so `apt` will know where to download the packages. Issue the following command to create a list file for MongoDB.

```
$ echo "deb [ arch=amd64,arm64 ] http://repo.mongodb.org/apt/ubuntu xenial/mongodb-org/3.4 mu
```

Finally, we'll update the packages list.

```
$ sudo apt-get update
```

Now we're ready to install MongoDB.

Step 2 — Installing MongoDB

We'll install the `mongodb-org` meta-package, which includes the daemon, configuration and init scripts, shell, and management tools on the server.

```
$ sudo apt-get install mongodb-org
```

Press enter or type `Y` to proceed when prompted. Once the installation is complete, we'll start the Mongo daemon:

```
$ sudo systemctl start mongod
```

Since `systemctl` doesn't provide output, we'll check the status to verify that the service has started properly.

```
$ sudo systemctl status mongod
```

Output

```
● mongod.service - High-performance, schema-free document-oriented database
   Loaded: loaded (/lib/systemd/system/mongod.service; disabled; vendor preset: enabled)
   Active: active (running) since Fri 2017-02-17 18:57:26 UTC; 17min ago
     Docs: https://docs.mongodb.org/manual
 Main PID: 2811 (mongod)
    Tasks: 17
   Memory: 56.8M
      CPU: 7.294s
   CGroup: /system.slice/mongod.service
           └─2811 /usr/bin/mongod --quiet --config /etc/mongod.conf
```

Press `q` to exit. Now that we've manually started the daemon and verified that it's running, we'll ensure that it restarts automatically at boot:

```
$ sudo systemctl enable mongod
```

The following output confirms that the command was successful:

```
Output
Created symlink from /etc/systemd/system/multi-user.target.wants/mongod.service
to /lib/systemd/system/mongod.service.
```

Next, we'll take essential steps to secure our databases.

Part Two: Securing MongoDB

Earlier versions of MongoDB were vulnerable to automated exploits because by default no authentication was required to interact with the database. Any user could create and destroy databases, as well as read from and write to their contents by default. This was compounded because those earlier versions also configured the MongoDB daemon to listen on all interfaces by default, which meant that automated scripts could detect MongoDB instances that weren't protected by a firewall and, if authentication hadn't been enabled, gain complete access to MongoDB.

The situation has been mitigated in the 3.x release as well as earlier versions provided by some package managers because the daemon is now bound to 127.0.0.1 so it will only accept connections on the Unix socket. It is not automatically open to the Internet.

However, authentication is still disabled by default, so any users on the local system have complete access to the databases. To secure this we'll create an administrative user, enable authentication and test.

Step 1 — Adding an Administrative User

To add our user, we'll connect to the Mongo shell:

```
$ mongo
```

The output when we use the Mongo shell warns us that access control is not enabled for the database and that read/write access to data and configuration is unrestricted.

```
Output
MongoDB shell version v3.4.2
connecting to: mongodb://127.0.0.1:27017
MongoDB server version: 3.4.2
Welcome to the MongoDB shell.
For interactive help, type "help".
For more comprehensive documentation, see
        http://docs.mongodb.org/
Questions? Try the support group
```

```
        http://groups.google.com/group/mongodb-user
Server has startup warnings:
2017-02-21T19:10:42.446+0000 I STORAGE  [initandlisten]
2017-02-21T19:10:42.446+0000 I STORAGE  [initandlisten] ** WARNING: Using the XFS filesystem i
2017-02-21T19:10:42.446+0000 I STORAGE  [initandlisten] **          See http://dochub.mongodb.
2017-02-21T19:10:42.534+0000 I CONTROL  [initandlisten]
2017-02-21T19:10:42.534+0000 I CONTROL  [initandlisten] ** WARNING: Access control is not enab
2017-02-21T19:10:42.534+0000 I CONTROL  [initandlisten] **          Read and write access to d
2017-02-21T19:10:42.534+0000 I CONTROL  [initandlisten]
>
```

We're free to choose the name for the administrative user since the privilege level comes from the assignment of the role `userAdminAnyDatabase`. The database, `admin` designates where the credentials are stored. You can learn more about authentication in the MongoDB Security Authentication section.

Set the username of your choice and be sure to pick your own secure password and substitute them in the command below:

```
> use admin
> db.createUser(
>   {
>     user: "AdminSammy",
>     pwd: "AdminSammy'sSecurePassword",
>     roles: [ { role: "userAdminAnyDatabase", db: "admin" } ]
>   }
> )
```

When we issue the `db.createUser` command, the shell will prepend three dots before each line until the command is complete. After that, we should receive feedback like the following when the user has been added.

Output

```
> use admin
switched to db admin
> db.createUser(
...   {
...     user: "AdminSammy",
...     pwd: "AdminSammy'sSecurePassword",
...     roles: [ { role: "userAdminAnyDatabase", db: "admin" } ]
```

```
...   }
... )
Successfully added user: {
        "user" : "AdminSammy",
        "roles" : [
                {
                        "role" : "userAdminAnyDatabase",
                        "db" : "admin"
                }
        ]
}
```

Type 'exit' and press `ENTER` or use `CTRL+C` to leave the client.

At this point, our user will be allowed to enter credentials, but they will not be **required** to do so until we enable authentication and restart the MongoDB daemon.

Step 2 — Enabling Authentication

Authentication is enabled in the `mongod.conf` file. Once we enable it and restart `mongod`, users still will be able to connect to Mongo without authenticating, but they will be required to provide a username and password before they can interact.

Let's open the configuration file:

```
$ sudo nano /etc/mongod.conf
```

In the `#security` section, we'll remove the hash in front of `security` to enable the stanza. Then we'll add the authorization setting. When we're done, the lines should look like the excerpt below:

mongodb.conf

```
 . . .
security:
  authorization: "enabled"
 . . .
```

Note that the “security” line has no spaces at the beginning, and the “authorization” line must be indented with two spaces

Once we've saved and exited the file, we'll restart the daemon:

```
$ sudo systemctl restart mongod
```

If we've made an error in the configuration, the dameon won't start. Since `systemctl` doesn't provide output, we'll use its `status` option to be sure that it did:.

```
$ sudo systemctl status mongod
```

If we see `Active: active (running)` in the output and it ends with something like the text below, we can be sure the `restart` command was successful:

```
Output
Jan 23 19:15:42 MongoHost systemd[1]: Started High-performance, schema-free document-oriented
```

Having verified the daemon is up, let's test authentication.

Step 3 — Verifying that Unauthenticated Users are Restricted

First, let's connect without credentials to verify that our actions are restricted:

```
$ mongo
```

Now that we've enabled authentication, all of the earlier warnings are resolved.

```
Output
MongoDB shell version v3.4.2
connecting to: mongodb://127.0.0.1:27017
MongoDB server version: 3.4.2
```

We're connected to the `test` database. We'll test that our access is restricted with the `show dbs` command:

```
> show dbs
```

```
Output
2017-02-21T19:20:42.919+0000 E QUERY    [thread1] Error: listDatabases failed:{
        "ok" : 0,
        "errmsg" : "not authorized on admin to execute command { listDatabases: 1.0 }",
        "code" : 13,
        "codeName" : "Unauthorized"
 . . .
```

We wouldn't be able to create users or similarily privileged tasks without authenticating.

Let's exit the shell to proceed:

```
> exit
```

Next, we'll make sure our Administrative user *does* have access.

Step 4 — Verifying the Administrative User's Access

We'll connect as our administrator with the `-u` option to supply a username and `-p` to be prompted for a password. We will also need to supply the database where we stored the user's authentication credentials with the `--authenticationDatabase` option.

```
$ mongo -u AdminSammy -p --authenticationDatabase admin
```

We'll be prompted for the password, so supply it. Once we enter the correct password, we'll be dropped into the shell, where we can issue the `show dbs` command:

```
Output
MongoDB shell version v3.4.2
Enter password:
connecting to: mongodb://127.0.0.1:27017
MongoDB server version: 3.4.2

>
```

Rather than being denied access, we should see the available databases:

```
> show dbs
```

```
Output
admin  0.000GB
local  0.000GB
```

Type `exit` or press `CTRL+C` to exit.

See the MongoDB documentation to learn more about Authentication, Role-Based Access Control, and Users and Roles.

Part Three: Configuring Remote Access (Optional)

Before we start working with an installation that allows remote connections, ideally we'll have MongoDB behind an external firewall, protected by a virtual private network (VPN), or restricted through a bastion host. As we work toward that, however, we can take the somewhat less-complicated step of enabling a firewall on the database server and restricting access to the specific host or hosts that need it.

Step 1 — Enabling UFW

In the Initial Server Setup with Ubuntu 16.04 prerequisite, we enabled UFW and allowed only SSH connections. Before we open a port for our client machine, let's verify UFW's status:

```
$ sudo ufw status
```

Note: If the output indicates that the firewall is `inactive`, activate it with:

```
host$ sudo ufw enable
```

Once it's enabled, rerunning the status command, `sudo ufw status` will show the rules. If necessary, be sure to allow SSH.

```
host$ sudo ufw allow OpenSSH
```

Unless we made changes to the prerequisites, the output should show that only OpenSSH is allowed:

```
Output
Status: active

To                         Action      From
--                         ------      ----
OpenSSH                    ALLOW       Anywhere
OpenSSH (v6)               ALLOW       Anywhere (v6)
```

Next, we'll allow access to the default MongoDB port, 27017, but restrict that access to a specific host. If you've changed the default port, be sure to update it in the command below.

```
host$ sudo ufw allow from client_ip_address to any port 27017
```

Re-run this command using the IP address for each additional client that needs access. To double-check the rule, we'll run `ufw status` again:

```
host$ sudo ufw status
```

```
Output
To                         Action      From
--                         ------      ----
OpenSSH                    ALLOW       Anywhere
27017                       ALLOW      client_ip_address
OpenSSH (v6)               ALLOW       Anywhere (v6)
```

> **Note:** If you're new to UFW, you can learn more in the guide, UFW Essentials: Common Firewall Rules and Commands.

With this firewall rule in place, we're ready to configure MongoDB to listen on its public interface.

Step 2 — Configuring a Public bindIP

To allow remote connections, we will add our host's publically-routable IP address to the `mongod.conf` file.

```
$ sudo nano /etc/mongod.conf
```

In the `net` stanza, add the `MongoHost` 's IP to the `bindIp` line:

Excerpt of /etc/mongod.conf

```
 . . .
net:
  port: 27017
  bindIp: 127.0.0.1,IP_of_MongoHost
 . . .
```

We'll save and exit the file, then restart the daemon:

```
$ sudo systemctl restart mongod
```

As we did earlier, we'll confirm restart was successful:

```
$ sudo systemctl status mongod
```

The output should contain `Active: active (running)` , and we can proceed to our final test. Mongo is now listening on its default port.

Step 3 — Testing the Remote Connection

We'll test that Mongo is listening on its public interface by adding the `--host` flag with the IP address from the `mongodb.conf` file.

```
$ mongo -u AdminSammy -p --authenticationDatabase admin --host IP_address_of_MongoHost
```

```
MongoDB shell version v3.4.2
Enter password:
```

```
connecting to: mongodb://107.170.233.82:27017/
MongoDB server version: 3.4.2
```

Reaching the prompt confirms that the daemon is listening on its public IP. At this point, any transaction between a remote connection and the MongoDB host is unencrypted so the next step, before testing the firewall, should be to secure those transations. For help with this, see MongoDB's Security documentation on Transport Encryption.

Conclusion

In this tutorial, we've added the MongoDB repository to our package list in order to install the latest available version of MongoDB, added an administrative user, and enabled authentication.

We've also shown how to configure MongoDB to accept remote connections but prevent advertising the MongoDB installation by configuring the server's firewall to allow connections only from hosts that require access.

Next Steps:

- To encrypt data in transit, see the MongoDB's Security documentation on Transport Encryption
- Learn more about using and administering MongoDB in these DigitalOcean community articles.

By: Melissa Anderson

Related Tutorials

How To Securely Configure a Production MongoDB Server

How to Install MongoDB on Ubuntu 18.04

How To Set Up Scheduled MongoDB Backups to DigitalOcean Spaces

How To Sync Transformed Data from MongoDB to Elasticsearch with Transporter on Ubuntu 16.04

Solution Deep Dive: Building a Highly Available Web Application with Web Processing and Stor...

16 Comments

Leave a comment...

Log In to Comment

lektin *March 4, 2017*

2 In part #2, after running `sudo systemctl status mongodb`, the result is like below,

```
● mongodb.service
   Loaded: not-found (Reason: No such file or directory)
   Active: inactive (dead)
```

Which say `Active: inactive (dead)`, rather than `Active: active (running)` indicated in the article.

So instead I ran `sudo systemctl status mongod`, and I got this,

```
● mongod.service - High-performance, schema-free document-oriented database
   Loaded: loaded (/lib/systemd/system/mongod.service; enabled; vendor preset: enable
   Active: active (running) since Sat 2017-03-04 10:28:55 UTC; 1min 4s ago
     Docs: https://docs.mongodb.org/manual
 Main PID: 14998 (mongod)
    Tasks: 17
   Memory: 47.4M
      CPU: 1.113s
   CGroup: /system.slice/mongod.service
           └─14998 /usr/bin/mongod --quiet --config /etc/mongod.conf
```

MelissaAnderson MOD *March 7, 2017*

0 Thank you for taking time to let us know. We've fixed that error in the tutorial.

JanisOz *April 26, 2017*

1 @MelissaAnderson This type still hasn't been fixed in the tutorial. It still says `sudo systemctl status mongodb` instead of `sudo systemctl status mongod`.

blakehschwartz *March 6, 2017*

0 Hello, I'm attempting to follow along and getting the following output during step 2:

```sudo systemctl start mongod
Failed to start mongod.service: Unit mongod.service not found.
blakers757@ubuntu-1gb-sfo1-01:~$ sudo systemctl status mongod
● mongod.service
Loaded: not-found (Reason: No such file or directory)
Active: failed (Result: exit-code) since Mon 2017-03-06 01:07:05 UTC; 22min ago
Main PID: 10538 (code=exited, status=48)

Mar 06 01:07:05 ubuntu-1gb-sfo1-01 systemd[1]: Started High-performance, schema-free document-oriented database.
Mar 06 01:07:05 ubuntu-1gb-sfo1-01 systemd[1]: mongod.service: Main process exited, code=exited, status=48/n/a
Mar 06 01:07:05 ubuntu-1gb-sfo1-01 systemd[1]: mongod.service: Unit entered failed state.
Mar 06 01:07:05 ubuntu-1gb-sfo1-01 systemd[1]: mongod.service: Failed with result 'exit-code'.
Mar 06 01:10:42 ubuntu-1gb-sfo1-01 systemd[1]: Stopped High-performance, schema-free document-oriented database.
Mar 06 01:21:36 ubuntu-1gb-sfo1-01 systemd[1]: Stopped mongod.service.
Mar 06 01:22:12 ubuntu-1gb-sfo1-01 systemd[1]: Stopped mongod.service.```

I tried uninstalling mongodb according to their official docs and reinstalling, still getting the same output. Any help would be great!

---

blakehschwartz *March 6, 2017*

1 Removing everything again, rebooting my server, and reinstalling solved my issue. I had started with an older tutorial on DO that had me install mongodb 3.2 and configure a systemd task, so even after I removed the file and re-installed mongodb I think it was still hung up somehow on the old installation. In any case, a reboot seemed to fix the issue.

---

MelissaAnderson MOD *March 7, 2017*

0 Thank you for taking time to report what the problem turned out to be!

---

BinhCAO *December 6, 2017*

0 Yes, you save my day!

---

ldaniellgoo *March 3, 2018*

0 i had the same problem i found the solution here no need to reboot u server https://stackoverflow.com/questions/37014186/running-mongodb-on-ubuntu-16-04-lts
```

lafimostafa *July 14, 2017*

0 Hello everyone I need your help I follow this tuto but I can not get to know myself from the outside either with **another terminal** or with **Robomongo**.

kouamx2017 *August 8, 2017*

0 Hello, I have a problem with my user admin

I create a user with role "root" over db admin, and I use this user in my application with NodeJS and mongoose.

When connect my application with MongoDB it is success, but when my application execute find or other operation show me a message error how "user not authorized to execute this command"

Can you help me please?

this is my uri to connect MongoDB
uri: mongodb://userinmongo:blablala@localhost:27017/dbname

this is the options to connect
options: {
auth: { authdb:"admin" }
server: { socketOptions: { keepAlive: 1, connectTimeoutMS: 30000 } },
replset: { socketOptions: { keepAlive: 1, connectTimeoutMS : 30000 } }
}

Thank you

elijahprince73 *May 11, 2018*

0 Hey I am having the same issue, were you able to find a solution to this

88rsousa *September 21, 2017*

1 There's a typo on Part 2 : Step 2

sudo systemctl status mongodb => should be => sudo systemctl status mongod

Great guide btw.

Thanks

MelissaAnderson MOD *September 21, 2017*

0 Thank you for taking time to point that out. We've made the correction.

dbsecurity *October 14, 2017*

0 Database security should include things like database firewall, data auditing for compliance (DAM) and data obfuscation in real time. Datasunrise to secure mongodb? datasunrise.com/mongodb/

mmonroy *February 24, 2018*

0 Thanks for the great tut.
After adding the mongo public ip to the bindIp separated by "," the service does not want to start. any idea why? if I remove the added ip everything works fine

vampiire *July 19, 2018*

0 when you type:

> sudo systemctl start mongod
> you get error:
> Failed to start mongod.service: Unit mongod.service not found.

to fix

> sudo systemctl enable mongod
> sudo service mongod restart
> sudo systemctl start mongod

now it works