MongoDB Installation in Ubntu

Step 1 — Installing MongoDB

Ubuntu's official package repositories include an up-to-date version of MongoDB, which means we can install the necessary packages using apt.

First, update the packages list to have the most recent version of the repository listings:

```
$ sudo apt update
```

Now install the MongoDB package itself:

```
$ sudo apt install -y mongodb
```

This command installs several packages containing the latest stable version of MongoDB, along with helpful management tools for the MongoDB server. The database server is automatically started after installation.

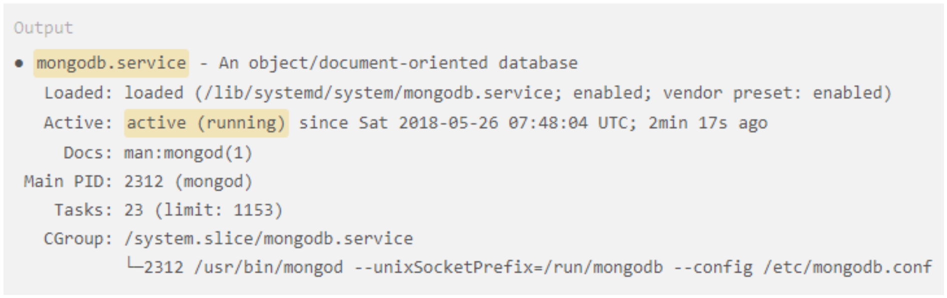
Next, let's verify that the server is running and works correctly.

Step 2 — Checking the Service and Database

The installation process started MongoDB automatically, but let's verifies that the service is started and that the database is working.First, check the service's status:

```
$ sudo systemctl status mongodb
```

You'll see this output:



```
Output
● mongodb.service - An object/document-oriented database
   Loaded: loaded (/lib/systemd/system/mongodb.service; enabled; vendor preset: enabled)
   Active: active (running) since Sat 2018-05-26 07:48:04 UTC; 2min 17s ago
     Docs: man:mongod(1)
 Main PID: 2312 (mongod)
    Tasks: 23 (limit: 1153)
   CGroup: /system.slice/mongodb.service
           └─2312 /usr/bin/mongod --unixSocketPrefix=/run/mongodb --config /etc/mongodb.conf
```

According to systemd, the MongoDB server is up and running.

We can verify this further by actually connecting to the database server and executing a diagnostic command

Execute this command:

```
$ mongo --eval 'db.runCommand({ connectionStatus: 1 })'
```

This will output the current database version, the server address and port, and the output of the status command:

```
Output
MongoDB shell version v3.6.3
connecting to: mongodb://127.0.0.1:27017
MongoDB server version: 3.6.3
{
        "authInfo" : {
                "authenticatedUsers" : [ ],
                "authenticatedUserRoles" : [ ]
        },
        "ok" : 1
}
```

A value of 1 for the ok field in the response indicates that the server is working properly.

Next, we'll look at how to manage the server instance.

Step 3 — Managing the MongoDB Service

MongoDB installs as a systemd service, which means that you can manage it using standard systemd commands alongside all other sytem services in Ubuntu.

To verify the status of the service, type:

```
$ sudo systemctl status mongodb
```

You can stop the server anytime by typing:

```
$ sudo systemctl stop mongodb
```

To start the server when it is stopped, type:

```
$ sudo systemctl start mongodb
```

You can also restart the server with a single command:

```
$ sudo systemctl restart mongodb
```

By default, MongoDB is configured to start automatically with the server. If you wish to disable the automatic startup, type:

```
$ sudo systemctl disable mongodb
```

It's just as easy to enable it again. To do this, use:

```
$ sudo systemctl enable mongodb
```

Next, let's adjust the firewall settings for our MongoDB installation.

Step 4 — Adjusting the Firewall (Optional)

Assuming you have followed the initial server setup tutorial instructions to enable the firewall on your server, the MongoDB server will be inaccessible from the internet.

If you intend to use the MongoDB server only locally with applications running on the same server, this is the recommended and secure setting. However, if you would like to be able to connect to your MongoDB server from the internet, you have to allow the incoming connections in ufw.

To allow access to MongoDB on its default port 27017 from everywhere, you could use sudo ufw allow 27017. However, enabling internet access to MongoDB server on a default installation gives anyone unrestricted access to the database server and its data.

In most cases, MongoDB should be accessed only from certain trusted locations, such as another server hosting an application. To accomplish this task, you can allow access on MongoDB's default port while specifying the IP address of another server that will be explicitly allowed to connect:

```
$ sudo ufw allow from your_other_server_ip/32 to any port 27017
```

You can verify the change in firewall settings with ufw:

```
$ sudo ufw status
```

You should see traffic to port 27017 allowed in the output:

```
Output

Status: active

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

If you have decided to allow only a certain IP address to connect to MongoDB server, the IP address of the allowed location will be listed instead of *Anywhere* in the output.

You can find more advanced firewall settings for restricting access to services in UFW Essentials: Common Firewall Rules and Commands.

Even though the port is open, MongoDB is currently only listening on the local address 127.0.0.1. To allow remote connections, add your server's publicly-routable IP address to the mongod.conf file.

Open the MongoDB configuration file in your editor:

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

Add your server's IP address to the bindIP value:

```
...
logappend=true

bind_ip = 127.0.0.1,your_server_ip
#port = 27017

...
```

Be sure to place a comma between the existing IP address and the one you added.

Save the file, exit the editor, and restart MongoDB:

```
$ sudo systemctl restart mongodb
```

MongoDB is now listening for remote connections, but anyone can access it.