Rocky Linux Middlewares Guide (English version)

A book from the Documentation Team

Version : 2024/03/20

Rocky Documentation Team

Table of contents

1. MariaDB database server

1.1 Prerequisites

- A Rocky Linux server
- Proficiency with a command-line editor (using *vi* in this example)
- A heavy comfort level with issuing commands from the command-line, viewing logs, and other general systems administrator duties
- An understanding of *mariadb-server* databases is helpful
- Run all commands as root or with *sudo*

1.2 Introduction

The *mariadb-server* and it is client *mariadb* are the open source alternatives to *mysql-server* and *mysql*, and they share command structure. *mariadb-server* is running on many web servers, due to the popular Wordpress CMS which requires it. This database, though, has many other uses.

If you want to use this along with other tools for hardening a web server, refer back to the Apache Hardened Web Server guide.

1.3 Installing `mariadb-server`

You need to install *mariadb-server*:

```
dnf install mariadb-server
```

1.4 Securing `mariadb-server`

To strengthen the security of *mariadb-server* you need to run a script, but before you do, you need to enable and start mariadb:

```
systemctl enable mariadb
```

Then:

`systemctl start mariadb`

Next, run this command:

`mysql_secure_installation`

> **Tip**
>
> The version of mariadb-server that comes enabled by default in Rocky Linux 8.5 is 10.3.32. You can install 10.5.13 by enabling the module:
>
> ```
> dnf module enable mariadb:10.5
> ```
>
> And then installing `mariadb`. As of version 10.4.6 of MariaDB, MariaDB specific commands are available that you can use instead of the old `mysql` prefixed commands. These include the previously mentioned `mysql_secure_installation` which can now be called with the MariaDB version `mariadb-secure-installation`.

This brings up a dialog:

```
NOTE: RUNNING ALL PARTS OF THIS SCRIPT IS RECOMMENDED FOR ALL MariaDB
      SERVERS IN PRODUCTION USE!  PLEASE READ EACH STEP CAREFULLY!

In order to log into MariaDB to secure it, you will need the current
password for the root user.  If you have just installed MariaDB, and
you have not set the root password yet, the password will be blank,
so you should just press enter here.

Enter current password for root (enter for none):
```

Since this is a brand-new installation, no root password set. Just hit enter here.

The next part of the dialog continues:

```
OK, successfully used password, moving on...

Setting the root password ensures that nobody can log into the MariaDB
root user without the proper authorisation.

Set root password? [Y/n]
```

You absolutely *do* want to have a root password set. You'll want to figure out what this should be and document it in a password manager somewhere so that you can pull it up if necessary. Start by hitting 'Enter' to accept the default "Y". This will bring up the password dialog:

```
New password:
Re-enter new password:
```

Enter your chosen password and then confirm it by entering it again. If this is successful, you will get the following dialog:

```
Password updated successfully!
Reloading privilege tables..
 ... Success!
```

Next the dialog deals with the anonymous user:

```
By default, a MariaDB installation has an anonymous user, allowing anyone
to log into MariaDB without having to have a user account created for
them.  This is intended only for testing, and to make the installation
go a bit smoother.  You should remove them before moving into a
production environment.

Remove anonymous users? [Y/n]
```

The answer here is "Y" so just hit 'Enter' to accept the default.

The dialog proceeds to the section dealing with allowing the root user to login remotely:

```
... Success!

Normally, root should only be allowed to connect from 'localhost'.  This
ensures that someone cannot guess at the root password from the network.

Disallow root login remotely? [Y/n]
```

root should only be needed locally on the machine. So accept this default as well by hitting 'Enter'.

The dialog then moves on to the 'test' database that is automatically installed with *mariadb-server*:

```
... Success!


By default, MariaDB comes with a database named 'test' that anyone can
```

```
access.  This is also intended only for testing, and should be removed
before moving into a production environment.

Remove test database and access to it? [Y/n]
```

Again, the answer here is the default, so just hit 'Enter' to remove it.

Finally, the dialog asks you if you want to reload the privileges:

```
- Dropping test database...
 ... Success!
 - Removing privileges on test database...
 ... Success!

Reloading the privilege tables will ensure that all changes made so far
will take effect immediately.

Reload privilege tables now? [Y/n]
```

Again, hit 'Enter' to do this. If all goes well, you will receive this message:

```
 ... Success!

Cleaning up...

All done!  If you've completed all of the above steps, your MariaDB
installation should now be secure.

Thanks for using MariaDB!
```

MariaDB will now be ready to use.

1.4.1 Rocky 9.0 changes

Rocky Linux 9.0 uses `mariadb-server-10.5.13-2` as the default mariadb-server version. As of version 10.4.3, a new plugin is automatically enabled in the server which changes the `mariadb-secure-installation` dialog. That plugin is `unix-socket` authentication. This article explains the new feature well. Essentially, using unix-socket authentication uses the logged-in user's credentials to access the database. It makes it so that if the root user, for example, logs in and then uses `mysqladmin` to create or delete a database (or any other function) that no password is needed for access. Same works with `mysql`. It also means there is no password to compromise

remotely. This depends on the security of the users set up on the server for all of the database protection.

The second dialog during the `mariadb-secure-installation` after the password is set for the administrative user is:

```
Switch to unix_socket authentication Y/n
```

The default here is "Y", but even if you answer "n", with the plugin enabled, a password is not requested for the user, at least not from the command line interface. You can specify either password or no password and they both work:

```
mysql

MariaDB [(none)]>
```

```
mysql -p
Enter password:

MariaDB [(none)]>
```

For more information on this feature, refer to the link above. There is a way to switch off this plugin and go back to having the password as a required field, which is also detailed within that link.

1.5 Conclusion

A database server, such as *mariadb-server*, can be used for many purposes. Because of the popularity of the Wordpress CMS, it is often found on web servers. Before you run the database in production, however, it is a good idea to strengthen its security.

2. Private DNS server with `bind`

2.1 Prerequisites and assumptions

- A server running Rocky Linux
- Several internal servers that need only local access, not over the Internet
- Several workstations that need access to these same servers that exist on the same network
- A healthy comfort level with entering commands from command line
- Familiarity with a command line editor (using *vi* in this example)
- Able to use *firewalld* for creating firewall rules

2.2 Introduction

External, or public, DNS servers map hostnames to IP addresses and, in the case of PTR (known as "pointer" or "reverse") records, map the IP addresses to the hostname. This is an essential part of the Internet. It makes your mail server, web server, FTP server, or many other servers and services work as expected no matter where you are.

On a private network, particularly one for developing many systems, you can use your Rocky Linux workstation's */etc/hosts* file to map a name to an IP address.

This will work for *your* workstation, but not for any other machine on your network. The best method to make things universally applied is to take some time out and create a local, private DNS server to handle this for all your machines.

Suppose you were creating production-level public DNS servers and resolvers. In that case, this author recommends the more robust PowerDNS authoritative and recursive DNS, which is installable on Rocky Linux servers. However, this document is for a local network that will not expose its DNS servers to the outside world. That is why the author chose `bind` for this example.

2.2.1 The DNS server components explained

DNS separates services into authoritative and recursive servers. These services are now recommended to be separate on separate hardware or containers.

The authoritative server is the storage area for all IP addresses and host names, and the recursive server looks up addresses and host names. In the case of our private DNS server, the authoritative and the recursive server services will run together.

2.3 Installing and enabling `bind`

The first step is to install packages:

```
dnf install bind bind-utils
```

bind's service daemon is `named`. Enable this to start on boot:

```
systemctl enable named
```

Start `named`:

```
systemctl start named
```

2.4 Configuration

Before making changes to any configuration file, create a backup copy of the original installed working file, *named.conf*:

```
cp /etc/named.conf /etc/named.conf.orig
```

That will help in the future if the introduction of errors into the configuration file occurs. It is *always* a good idea to make a backup copy before making changes.

Edit the *named.conf* file. The author is using *vi* , but you can substitute your favorite command line editor:

```
vi /etc/named.conf
```

Turn off listening on the localhost. Do this by remarking with a "#" sign, these two lines in the "options" section. This shuts down any connection to the outside world.

This is helpful, particularly when you add this DNS to our workstations because you want the DNS server to only respond when the IP address requesting the service is local and not react if the server or service is on the Internet.

This way, the other configured DNS servers will take over nearly immediately to look up the Internet based services:

```
options {
#       listen-on port 53 { 127.0.0.1; };
#       listen-on-v6 port 53 { ::1; };
```

Finally, skip down to the bottom of the *named.conf* file and add a section for your network. Our example is "ourdomain", so sub in what you want to call your LAN hosts:

```
# primary forward and reverse zones
//forward zone
zone "ourdomain.lan" IN {
     type master;
     file "ourdomain.lan.db";
     allow-update { none; };
    allow-query {any; };
};
//reverse zone
zone "1.168.192.in-addr.arpa" IN {
     type master;
     file "ourdomain.lan.rev";
     allow-update { none; };
    allow-query { any; };
};
```

Save your changes (for *vi*, `SHIFT:wq!`)

2.5 The forward and reverse records

You need to create two files in `/var/named`. You will edit these files if you add machines to your network to include them in the DNS.

The first is the forward file to map our IP address to the hostname. Again, our examples is "ourdomain" here. Note that the IP of our local DNS is 192.168.1.136. Add hosts at the bottom of this file.

```
vi /var/named/ourdomain.lan.db
```

The file will look something like this when completed:

```
$TTL 86400
@ IN SOA dns-primary.ourdomain.lan. admin.ourdomain.lan. (
    2019061800 ;Serial
    3600 ;Refresh
    1800 ;Retry
    604800 ;Expire
    86400 ;Minimum TTL
)

;Name Server Information
@ IN NS dns-primary.ourdomain.lan.

;IP for Name Server
dns-primary IN A 192.168.1.136

;A Record for IP address to Hostname
wiki IN A 192.168.1.13
www IN A 192.168.1.14
devel IN A 192.168.1.15
```

Add all the hosts and IP addresses you need and save your changes.

You need a reverse file to map our hostname to the IP address. In this case, the only part of the IP that you need is the last octet (in an IPv4 address each number separated by a "." is an octet) of the host, the PTR, and hostname.

```
vi /var/named/ourdomain.lan.rev
```

When completed, the file will look something like this:

```
$TTL 86400
@ IN SOA dns-primary.ourdomain.lan. admin.ourdomain.lan. (
    2019061800 ;Serial
    3600 ;Refresh
    1800 ;Retry
    604800 ;Expire
    86400 ;Minimum TTL
)
;Name Server Information
@ IN NS dns-primary.ourdomain.lan.

;Reverse lookup for Name Server
136 IN PTR dns-primary.ourdomain.lan.

;PTR Record IP address to HostName
13 IN PTR wiki.ourdomain.lan.
14 IN PTR www.ourdomain.lan.
15 IN PTR devel.ourdomain.lan.
```

Add all of the hostnames that are in the forward file and save your changes.

2.5.1 What all this means

Since you have all of this added in, and are preparing to restart our *bind* DNS server, let us explore some of the terminologies used in these two files.

Just making things work is not good enough if you do not know what each term means, right?

- **TTL** stands for "Time To Live". TTL tells the DNS server how long to keep its cache before requesting a fresh copy. In this case, the TTL is the default setting for all records unless you manually enter a specific TTL. The default here is 86400 seconds or 24 hours.
- **IN** stands for Internet. In this case, the Internet is not used. Think of this as the Intranet instead.
- **SOA** stands for "Start Of Authority" or what the primary DNS server is for the domain
- **NS** stands for "name server"
- **Serial** is the value used by the DNS server to verify that the contents of the zone file are up-to-date
- **Refresh** specifies how often a slave DNS server will request a zone transfer from the master
- **Retry** specifies the length of time in seconds to wait before trying again on a failed zone transfer
- **Expire** specifies how long a slave server will wait to answer a query when the master is unreachable
- **A** Is the host address or forward record and is only in the forward file
- **PTR** The pointer record better known as the "reverse" and is only in our reverse file

2.6 Testing configurations

When you have all of your files created, you need to ensure that the configuration files and zones are in good working order before you start the *bind* service again.

Check the main configuration:

```
named-checkconf
```

This will return an empty result if everything is OK.

Check the forward zone:

```
named-checkzone ourdomain.lan /var/named/ourdomain.lan.db
```

This will return something like this if all is well:

```
zone ourdomain.lan/IN: loaded serial 2019061800
OK
```

Finally, check the reverse zone:

```
named-checkzone 192.168.1.136 /var/named/ourdomain.lan.rev
```

Which will return something like this if all is well:

```
zone 192.168.1.136/IN: loaded serial 2019061800
OK
```

Assuming that everything looks good, go ahead and restart *bind*:

```
systemctl restart named
```

9

2.7 9 using IPv4 on your LAN

To use ONLY IPv4 on your LAN, you need to make one change in `/etc/sysconfig/named`:

```
vi /etc/sysconfig/named
```

Add this at the bottom of the file:

```
OPTIONS="-4"
```

Save those changes.

2.8 9 Testing machines

You need to add the DNS server (in our example 192.168.1.136) to each machine that you want to have access to the servers that you added to your local DNS. The author only shows an example of how to do this on a Rocky Linux workstation. Similar methods exist for other Linux distributions, Windows, and Mac machines.

You will want to add the DNS servers to the list, not replace what is currently there, as you will still need Internet access, which will require your presently assigned DNS servers. DHCP (Dynamic Host Configuration Protocol) services generally assign these or they are statically assigned.

You will add our local DNS with `nmcli` and then restart the connection.

Stupid Profile Names

In NetworkManager, the connections are not modified by the name of the device but by the name of the profile. This can be things like "Wired connection 1" or "Wireless connection 1". You can see the profile by running `nmcli` without any parameters:

```
nmcli
```

This will show you output such as:

```
enp0s3: connected to Wired Connection 1
"Intel 82540EM"
ethernet (e1000), 08:00:27:E4:2D:3D, hw, mtu 1500
ip4 default
inet4 192.168.1.140/24
route4 192.168.1.0/24 metric 100
route4 default via 192.168.1.1 metric 100
inet6 fe80::f511:a91b:90b:d9b9/64
route6 fe80::/64 metric 1024

lo: unmanaged
    "lo"
    loopback (unknown), 00:00:00:00:00:00, sw, mtu 65536

DNS configuration:
    servers: 192.168.1.1
```