Rocky Linux Web Services Guide (English version)

A book from the Documentation Team

Version : 2024/07/08

Rocky Documentation Team

Table of contents

1. Foreword

Rocky Linux is part of the Enterprise Linux family, making it particularly well suited to hosting web services such as file servers (FTP, sFTP), web servers (apache, nginx), application servers (PHP, Python), database servers (MariaDB, Mysql, PostgreSQL) or more specific services such as load balancing, caching, proxy or reverse proxy (HAProxy, Varnish, Squid).

The web would not be what it is without email. Web services generally make extensive use of mail servers (Postfix).

Sometimes these services are extremely busy or require highly available services. In these cases, other services can be implemented to guarantee optimal service performance (Heartbeat, PCS).

Each chapter of this book can be consulted independently, according to your needs, and it is not compulsory to read the chapters in order.

This book is also part of a series of books dedicated to system administration under Linux (Admin Guide, Learning Bash, Learning Ansible). Where necessary, you will be invited to review the concepts you may be missing in the corresponding chapters of the above-mentioned books.

1.1 Public

The target audience for this book is system administrators already trained in the use of system administration commands (see our book Admin Guide), who want to install, configure and secure their web services.

1.2 How to use this book

This book has been designed as a training manual, so that it can be used in several ways. Either as a training aid for trainers, or as a self-training aid for administrators wishing to acquire new skills or reinforce their existing knowledge.

To implement some of the services presented in this book, you may need two (or more) servers to put the theory into practice.

2. Licence

RockyLinux offers Linux courseware for trainers or people wishing to learn how to administer a Linux system on their own.

RockyLinux materials are published under Creative Commons-BY-SA. This means you are free to share and transform the material, while respecting the author's rights.

BY : **Attribution**. You must cite the name of the original author.

SA : **Share Alike**.

- Creative Commons-BY-SA licence : https://creativecommons.org/licenses/by-sa/4.0/

The documents and their sources are freely downloadable from:

- https://docs.rockylinux.org
- https://github.com/rocky-linux/documentation

Our media sources are hosted at github.com. You'll find the source code repository where the version of this document was created.

From these sources, you can generate your own personalized training material using mkdocs. You will find instructions for generating your document here.

How can I contribute to the documentation project?

You'll find all the information you need to join us on our git project home page.

We wish you all a pleasant reading and hope you enjoy the content.

3. Part 1. Files Servers

4. Part 2. Web Servers

4.1 Introduction

4.1.1 HTTP Protocol

HTTP (**H**yper**T**ext **T**ransfer **P**rotocol) has been the most widely used protocol on the Internet since 1990.

This protocol enables the transfer of files (mainly in HTML format, but also in CSS, JS, AVI...) localized by a character string called **URL** between a browser (the client) and a Web server (called `httpd` on UNIX machines).

HTTP is a "request-response" protocol operating on top of **TCP** (**T**ransmission **C**ontrol **P**rotocol).

1. The client opens a TCP connection to the server and sends a request.
2. The server analyzes the request and responds according to its configuration.

The HTTP protocol is "**STATELESS**": it does not retain any information about the client's state from one request to the next. Dynamic languages such as php, python, or java store client session information in memory (as on an e-commerce site, for example).

The HTTP protocol is version 1.1. Version 2 is still under development.

An HTTP response is a set of lines sent to the browser by the server. It includes:

- A **status line**: this specifies the protocol version used and the processing status of the request, using a code and explanatory text. The line comprises three elements separated with a space:
- The protocol version used
- The status code
- The meaning of the code
- **Response header fields**: these are a set of optional lines providing additional information about the response and/or the server. Each of these lines consists of a name qualifying the header type, followed by a colon (:) and the header value.
- **The response body**: this contains the requested document.

Here is an example of an HTTP response:

```
$ curl --head --location https://docs.rockylinux.org
HTTP/2 200
accept-ranges: bytes
access-control-allow-origin: *
age: 109725
cache-control: public, max-age=0, must-revalidate
content-disposition: inline
content-type: text/html; charset=utf-8
date: Fri, 21 Jun 2024 12:05:24 GMT
etag: "cba6b533f892339d3818dc59c3a5a69a"
server: Vercel
strict-transport-security: max-age=63072000
x-vercel-cache: HIT
x-vercel-id: cdg1::pdqbh-1718971524213-4892bf82d7b2
content-length: 154696
```

Note

Learning the `curl` command usages will be very helpfull for you to troubleshoot your servers in the future.

The role of the web server is to translate a URL into a local resource. Consulting the https://docs.rockylinux.org/ page is like sending an HTTP request to this machine. The DNS service therefore plays an essential role.

4.1.2 URLs

A **URL** (**U**niform **R**esource **L**ocator) is an ASCII character string used to designate resources on the Internet. It is informally referred to as a web address.

A URL has three parts:

```
<protocol>://<host>:<port>/<path>
```

- **Protocol name**: this is the language used to communicate over the network, for example HTTP, HTTPS, FTP, and so on. The most widely used protocols are HTTP (HyperText TransferProtocol) and its secure version HTTPS, the protocol used to exchange Web pages in HTML format.
- **Login** and **password**: allows you to specify access parameters to a secure server. This option is not recommended, as the password is visible in the URL (for security purposes).
- **Host**: This is the name of the computer hosting the requested resource. Note that it is possible to use the server's IP address, which makes the URL less readable.
- **Port number**: this is a number associated with a service, enabling the server to know the requested resource type. The default port associated with the HTTP protocol is port number 80 and 443 with HTTPS. So, when the protocol in use is HTTP or HTTPS, the port number is optional.
- Resource path: This part lets the server know the location of the resource. Generally, the location (directory) and name of the requested file. If nothing in the address specifies a location, it indicates the first page of the host. Otherwise it indicates the path to the page to display.

4.1.3 Ports

An HTTP request will arrive on port 80 (default port for http) of the server running on the host. However, the administrator is free to choose the server's listening port.

The http protocol is available in a secure version: the https protocol (port 443). Implement this encrypted protocol with the `mod_ssl` module.

Using other ports is also possible, such as port `8080` (Java EE application servers).

4.2 Apache

In this chapter, you will learn about Apache, the web server.

Objectives: In this chapter, you will learn how to:

✔ install and configure apache

🏁 **apache**, **http**, **httpd**

Knowledge: ⭐ ⭐ **Complexity**: ⭐ ⭐

Reading time: 30 minutes

4.2.1 Generalities

The Apache HTTP server is the work of a group of volunteers: The Apache Group. This group set out to build a Web server on the same level as commercial products, but as free software (its source code is available).

Joining the original team were hundreds of users who, through their ideas, tests, and lines of code, contributed to making Apache the most widely used Web server in the world.

Apache's ancestor is the free server developed by the National Center for Supercomputing Applications at the University of Illinois. The evolution of this server came to a halt when the person in charge left the NCSA in 1994. Users continued to fix bugs and create extensions, which they distributed as "patches", hence the name "a patchee server".

The release of Apache version 1.0 was on December 1, 1995 (over 30 years ago!).

The development team coordinates its work by way of a mailing list, where discussions regarding proposals and changes to the software happen. Voting on changes happens before incorporation into the project. Anyone can join the development team: all you need to do to become a member of The Apache Group is make an active contribution to the project.

The Apache server has a very strong presence on the Internet, still accounting for around 50% of market share for all active sites.

The market share lost by Apache often goes to its biggest challenger: the nginx server. The latter is faster at delivering web pages, and less functionally complete than the giant Apache.

4.2.2 Installation

Apache is **cross-platform**. It is usable on Linux, Windows, Mac...

The administrator will have to choose between two installation methods:

- **Package installation**: the distribution vendor supplies **stable, supported** (but sometimes older) versions
- **Installation from source**: which involves compilation of the software by the administrator, who can specify the options that interest him or her, thus optimizing the service. Since Apache has a modular architecture, it is generally not necessary to re-compile the apache software to add or remove additional functionalities (add or remove modules).

The package-based installation method is strongly recommended. Additional repositories are available to install more recent versions of apache on older distributions, but nobody will provide support in the event of problems.

On Enterprise Linux distributions, the `httpd` package provides the Apache server.

In the future, you might have to install some extra modules. Here are some examples of modules and their roles:

- **mod_access**: filters client access by host name, IP address or other characteristic
- **mod_alias**: enables the creation of aliases or virtual directories
- **mod_auth**: authenticates clients
- **mod_cgi**: executes CGI scripts
- **mod_info**: provides information on server status
- **mod_mime**: associates file types with the corresponding action
- **mod_proxy**: proposes a proxy server
- **mod_rewrite**: rewrites URLs
- Others

```
sudo dnf install httpd
```

The version installed on Rocky Linux 9 is 2.4.

Installing the package creates an `apache` system user and a corresponding `apache` system group.

```
$ grep apache /etc/passwd
apache:x:48:48:Apache:/usr/share/httpd:/sbin/nologin
$ grep apache /etc/group
apache:x:48:
```

Enable and start the service:

```
$ sudo systemctl enable httpd --now
Created symlink /etc/systemd/system/multi-user.target.wants/httpd.service → /
usr/lib/systemd/system/httpd.service.
```

You can check the service's status:

```
$ sudo systemctl status httpd
● httpd.service - The Apache HTTP Server
     Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; preset:
disabl>     Active: active (running) since Fri 2024-06-21 14:22:34 CEST; 8s ago
```

```
        Docs: man:httpd.service(8)
    Main PID: 4387 (httpd)
      Status: "Started, listening on: port 80"
       Tasks: 177 (limit: 11110)
      Memory: 24.0M
         CPU: 68ms
      CGroup: /system.slice/httpd.service
              ├─4387 /usr/sbin/httpd -DFOREGROUND
              ├─4389 /usr/sbin/httpd -DFOREGROUND
              ├─4390 /usr/sbin/httpd -DFOREGROUND
              ├─4391 /usr/sbin/httpd -DFOREGROUND
```

Do not forget to open your firewall (see Security section).

You can check now the availability of the service:

- from any web browser providing the IP address of your server (for example http://192.168.1.100/).
- directly from your server.

For that, you will have to install a text browser, for example elinks.

```
sudo dnf install elinks
```

Browse your server and check the default page:

```
elinks http://localhost
```

Installing the `httpd` package generates a complete tree structure that needs to be fully understood:

```
/etc/httpd/
├── conf
│   ├── httpd.conf
│   └── magic
├── conf.d
│   ├── README
│   ├── autoindex.conf
│   ├── userdir.conf
│   └── welcome.conf
├── conf.modules.d
│   ├── 00-base.conf
│   ├── 00-brotli.conf
```

```
│   ├── 00-dav.conf
│   ├── 00-lua.conf
│   ├── 00-mpm.conf
│   ├── 00-optional.conf
│   ├── 00-proxy.conf
│   ├── 00-systemd.conf
│   ├── 01-cgi.conf
│   ├── 10-h2.conf
│   ├── 10-proxy_h2.conf
│   └── README
├── logs -> ../../var/log/httpd
├── modules -> ../../usr/lib64/httpd/modules
├── run -> /run/httpd
└── state -> ../../var/lib/httpd
/var/log/httpd/
├── access_log
└── error_log
/var/www/
├── cgi-bin
└── html
```

You will notice that the `/etc/httpd/logs` folder is a symbolic link to the `/var/log/httpd` directory. Similarly, you will notice that the files making up the default site are in the `/var/www/html` folder.

4.2.3 Configuration

Initially, configuration of the Apache server was in a single `/etc/httpd/conf/httpd.conf` file. Over time, this file has become increasingly large and less readable.

Modern distributions therefore tend to distribute Apache configuration over a series of `*.conf` files in the directories `/etc/httpd/conf.d` and `/etc/httpd/conf.modules.d`, attached to the main `/etc/httpd/conf/httpd.conf` file by the Include directive.

```
$ sudo grep "^Include" /etc/httpd/conf/httpd.conf
Include conf.modules.d/*.conf
IncludeOptional conf.d/*.conf
```

The `/etc/httpd/conf/httpd.conf` file is amply documented. In general, these comments are sufficient to clarify the administrator's options.

Global server configuration is in `/etc/httpd/conf/httpd.conf`.

This file has 3 sections for configuring:

- in **section 1**, the global environment;
- in **section 2**, the default site and default virtual site parameters;
- in **section 3**, the virtual hosts.

Virtual hosting lets you put **several virtual sites online** on the same server. The sites are then differentiated according to their domain names, IP addresses, and so on.

Modifying a value in section 1 or 2 affects all hosted sites.

In a shared environment, modifications are therefore in section 3.

To facilitate future updates, it is strongly recommended that you create a section 3 configuration file for each virtual site.

Here is a minimal version of the `httpd.conf` file:

```
ServerRoot "/etc/httpd"
Listen 80
Include conf.modules.d/*.conf
User apache
Group apache
ServerAdmin root@localhost
<Directory />
    AllowOverride none
    Require all denied
</Directory>
DocumentRoot "/var/www/html"
<Directory "/var/www">
    AllowOverride None
    Require all granted
</Directory>
<Directory "/var/www/html">
    Options Indexes FollowSymLinks
    AllowOverride None
    Require all granted
</Directory>
<IfModule dir_module>
    DirectoryIndex index.html
</IfModule>
```

```
<Files ".ht*">
    Require all denied
</Files>
ErrorLog "logs/error_log"
LogLevel warn
<IfModule log_config_module>
    LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\""
combined
    LogFormat "%h %l %u %t \"%r\" %>s %b" common
    <IfModule logio_module>
      LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"
%I %O" combinedio
    </IfModule>
    CustomLog "logs/access_log" combined
</IfModule>
<IfModule alias_module>
    ScriptAlias /cgi-bin/ "/var/www/cgi-bin/"
</IfModule>
<Directory "/var/www/cgi-bin">
    AllowOverride None
    Options None
    Require all granted
</Directory>
<IfModule mime_module>
    TypesConfig /etc/mime.types
    AddType application/x-compress .Z
    AddType application/x-gzip .gz .tgz
    AddType text/html .shtml
    AddOutputFilter INCLUDES .shtml
</IfModule>
AddDefaultCharset UTF-8
<IfModule mime_magic_module>
    MIMEMagicFile conf/magic
</IfModule>
EnableSendfile on
IncludeOptional conf.d/*.conf
```

Section 1

The various directives encountered in section 1 are :

Option	Information
`ServerTokens`	This directive will be in a future chapter.
`ServertRoot`	Indicates the path to the directory containing all the files making up the Apache server.
`Timeout`	The number of seconds before the expiry time of a too long request (incoming or outgoing).
`KeepAlive`	Persistent connection (several requests per TCP connection).
`MaxKeepAliveRequests`	Maximum number of persistent connections.
`KeepAliveTimeout`	Number of seconds to wait for the next client request before closing the TCP connection.
`Listen`	Allow apache to listen on specific addresses or ports.
`LoadModule`	Load add-on modules (fewer modules = greater security).
`Include`	Include other server configuration files.
`ExtendedStatus`	Display more information about the server in the server-status module.
`User` and `Group`	Allows the launching of Apache processes with different users. Apache always starts as root, then changes its owner and group.

MULTI-PROCESS MODULES (MPM)

The Apache server was designed to be a powerful and flexible server, capable of running on a wide variety of platforms.

Different platforms and environments often mean different functionality, or the use of different methods to implement the same functionality as efficiently as possible.

Apache's modular design allows the administrator to choose which features to include in the server, by selecting which modules to load, either at compile-time or at run-time.

This modularity also includes the most basic web server functions.

Certain modules, the Multi-Process Modules (MPM), are responsible for associating with the machine's network ports, accepting requests and distributing them among the various child processes.

Configuring MPM modules is in the `/etc/httpd/conf.modules.d/00-mpm.conf` configuration file:

```
# Select the MPM module which should be used by uncommenting exactly
# one of the following LoadModule lines.  See the httpd.conf(5) man
# page for more information on changing the MPM.

# prefork MPM: Implements a non-threaded, pre-forking web server
# See: http://httpd.apache.org/docs/2.4/mod/prefork.html
#
# NOTE: If enabling prefork, the httpd_graceful_shutdown SELinux
# boolean should be enabled, to allow graceful stop/shutdown.
#
#LoadModule mpm_prefork_module modules/mod_mpm_prefork.so

# worker MPM: Multi-Processing Module implementing a hybrid
# multi-threaded multi-process web server
# See: http://httpd.apache.org/docs/2.4/mod/worker.html
#
#LoadModule mpm_worker_module modules/mod_mpm_worker.so

# event MPM: A variant of the worker MPM with the goal of consuming
# threads only for connections with active processing
# See: http://httpd.apache.org/docs/2.4/mod/event.html
#
LoadModule mpm_event_module modules/mod_mpm_event.so
```

As you can see, the default MPM is the `mpm_event`.

The performance and capabilities of your web server depend heavily on the choice of MPM.

Choosing one module over another is therefore a complex task, as is optimizing the chosen MPM module (number of clients, queries, and so on.).

By default, the Apache configuration assumes a moderately busy service (256 clients max).

ABOUT KEEPALIVE DIRECTIVES

With the `KeepAlive` directive disabled, every resource request on the server requires opening a TCP connection, which is time-consuming from a network point of view and requires a lot of system resources.

With the `KeepAlive` directive set to `On`, the server keeps the connection open with the client for the duration of the `KeepAlive`.

Given that a web page contains several files (images, stylesheets, javascripts, etc.), this strategy is a quick winner.

However, it is important to set this value as precisely as possible:

- Too short a value penalizes the customer,
- Too long a value penalizes server resources.

`KeepAlive` values for individual customer virtual hosts allows more granularity per customer. In this case, setting `KeepAlive` values happens directly in the customer's VirtualHost or at proxy level (`ProxyKeepalive` and `ProxyKeepaliveTimeout`).

Section 2

Section 2 sets the values used by the main server. The main server responds to all requests that are not handled by one of the Virtualhosts in section 3.

The values are also used as default values for virtual sites.

Option	Information
`ServerAdmin`	specifies an e-mail address which will appear on certain auto-generated pages, such as error pages.
`ServerName`	specifies the name identifying the server. Can happen automatically, but it the recommendation is to specify it explicitly (IP address or DNS name).
`DocumentRoot`	specifies the directory containing files to serve to clients. Default /var/www/html/.
`ErrorLog`	specifies the path to the error file.
`LogLevel`	debug, info, notice, warn, error, crit, alert, emerg.
`LogFormat`	defines a specific log format. Done with the CustomLog directive.
`CustomLog`	specify path to access file.
`ServerSignature`	seen in the security part.
`Alias`	specifies a directory outside the tree and makes it accessible by context. The presence or absence of the last slash in the context is important.
`Directory`	specifies behaviors and access rights by directory.
`AddDefaultCharset`	specifies the encoding format for pages sent (accented characters can be replaced by ?...).
`ErrorDocument`	customized error pages.
`server-status`	report on server status.
`server-info`	report on server configuration.

THE `ERRORLOG` DIRECTIVE

The `ErrorLog` directive defines the error log to use.

This directive defines the name of the file in which the server logs all errors it encounters. If the file path is not absolute, the assumption is to be relative to ServerRoot.

THE `DIRECTORYINDEX` DIRECTIVE

The DirectoryIndex directive defines the site's home page.

This directive specifies the name of the file loaded first, which will act as the site index or home page.

Syntax:

```
DirectoryIndex display-page
```

The full path is not specified. Searching for the file happens in the directory specified by DocumentRoot.

Example:

```
DocumentRoot /var/www/html
DirectoryIndex index.php index.htm
```

This directive specifies the name of the website index file. The index is the default page that opens when the client types the site URL (without having to type the index name). This file must be in the directory specified by the `DocumentRoot` directive.

The `DirectoryIndex` directive can specify several index file names separated by spaces. For example, a default index page with dynamic content and, as a second choice, a static page.

THE `DIRECTORY` DIRECTIVE

The Directory tag is used to define directory-specific directives.

This tag applies rights to one or more directories. The directory path is entered as an absolute.

Syntax:

```
<Directory directory-path>
Defining user rights
</Directory>
```

Example:

```
<Directory /var/www/html/public>
    Require all granted   # we allow everyone
</Directory>
```

The `Directory` section defines a block of directives applying to a part of the server's file system. The directives contained here will only apply to the specified directory (and its sub-directories).

The syntax of this block accepts wildcards, but it is preferable to use the DirectoryMatch block.

In the following example, we're going to deny access to the server's local hard disk, regardless of the client. The "/" directory represents the root of the hard disk.

```
<Directory />
    Require all denied
</Directory>
```

The following example shows authorizing access to the /var/www/html publishing directory for all clients.

```
<Directory /var/www/html>
    Require all granted
</Directory>
```

When the server finds an `.htaccess` file, it needs to know whether directives placed in the file have authorization to modify the pre-existing configuration. The `AllowOverride` directive, controls that authorization in `Directory` directives. When set to `none`, `.htaccess` files are completely ignored.

THE `MOD_STATUS`

The `mod_status` displays a `/server-status` or `/server-info` page summarizing server status:

```
<Location /server-status>
    SetHandler server-status
    Require local
</Location>

<Location /server-info>
    SetHandler server-info
    Require local
</Location>
```

Please note that this module provides information that should not be accessible to your users.

Shared hosting (section 3)

With shared hosting, the customer thinks they are visiting several servers. In reality, there is just one server and several virtual sites.

To set up shared hosting, you need to set up virtual hosts:

• declaring multiple listening ports

• declaring multiple listening IP addresses (virtual hosting by IP)

• declaring multiple server names (virtual hosting by name)

Each virtual site corresponds to a different tree structure.

Section 3 of the `httpd.conf` file declares these virtual hosts.

To facilitate future updates, it is strongly recommended that you create a section 3 configuration file for each virtual site.

Choose virtual hosting "by IP" or "by name". For production use, it is not advisable to mix the two solutions.

- Configuring each virtual site in an independent configuration file
- VirtualHosts are stored in `/etc/httpd/conf.d/`
- The file extension is `.conf`

THE `VIRTUALHOST` DIRECTIVE

The `VirtualHost` directive defines virtual hosts.

```
<VirtualHost IP-address[:port]>
    # if the "NameVirtualHost" directive is present
    # then "address-IP" must match the one entered
    # under "NameVirtualHost" as well as for "port".
 ...
 </VirtualHost>
```

If you configure the Apache server with the basic directives seen above, you will only be able to publish one site. Indeed, you can not publish multiple sites with the default settings: same IP address, same TCP port and no hostname or unique hostname.

The use of virtual sites will enable us to publish several websites on the same Apache server. You are going to define blocks, each of which will describe a website. In this way, each site will have its own configuration.

For ease of understanding, a website is often associated with a single machine. Virtual sites or virtual hosts are so called because they dematerialize the link between machine and website.

Example 1:

```
Listen 192.168.0.10:8080
<VirtualHost 192.168.0.10:8080>
  DocumentRoot /var/www/site1/
  ErrorLog /var/log/httpd/site1-error.log
</VirtualHost>

Listen 192.168.0.11:9090
```

```
<VirtualHost 192.168.0.11:9090>
  DocumentRoot /var/www/site2/
  ErrorLog /var/log/httpd/site2-error.log
</VirtualHost>
```

IP-based virtual hosting is a method of applying certain guidelines based on the IP address and port on which the request is received. In general, this means serving different web sites on different ports or interfaces.

THE `NAMEVIRTUALHOST` DIRECTIVE

The `NameVirtualHost` directive defines name-based virtual hosts.

This directive is mandatory for setting up name-based virtual hosts. With this directive, you specify the IP address on which the server will receive requests from name-based virtual hosts.

Syntax:

```
NameVirtualHost adresse-IP[:port]
```

Example:

```
NameVirtualHost 160.210.169.6:80
```

The directive must come before the virtual site description blocks. It designates the IP addresses used to listen for client requests to virtual sites.

To listen for requests on all the server's IP addresses, use the * character.

Taking changes into account

For each configuration change, it is necessary to reload the configuration with the following command:

```
sudo systemctl reload httpd
```

Manual

There is a package containing a site that acts as an Apache user manual. It is called `httpd-manual`.

```
sudo dnf install httpd-manual
sudo systemctl reload httpd
```

When installed, you can access the manual with a web browser at http://127.0.0.1/manual.

```
$ elinks http://127.0.0.1/manual
```

The `apachectl` command

The `apachectl` is the server control interface for Apache `httpd` server.

It is a very usefull command with the `-t` or `configtest` witch run a configuration file syntax test.

> **Note**
>
> Very usefull when used with ansible handlers to test the configuration.

4.2.4 Security

When protecting your server with a firewall (which is a good thing), you might need to consider opening it.

```
sudo firewall-cmd --zone=public --add-service=http
sudo firewall-cmd --zone=public --add-service=https
sudo firewall-cmd --reload
```

SELinux

By default, if SELinux security is active, it prevents the reading of a site from a directory other than `/var/www/`.

The directory containing the site must have the security context `httpd_sys_content_t`.

You can check current context with the command:

```
* ls -Z /dir
```

Add context with the following command:

```
sudo chcon -vR --type=httpd_sys_content_t /dir
```

It also prevents the opening of a non-standard port. Opening the port is a manual operation, using the `semanage` command (not installed by default).

```
sudo semanage port -a -t http_port_t -p tcp 1664
```

User and Group directives

the `User` and `Group` directives define an Apache management account and group.

Historically, root ran Apache, which caused security problems. Apache is always run by root, but then changes its identity. Generally User `apache` and Group `apache`.

Never ROOT!

The Apache server (`httpd` process) starts with the `root` superuser account. Each client request triggers the creation of a "child" process. To limit risks, launching these child processes happens from a less privileged account.

The User and Group directives declare the account and group used to create child processes.

This account and group must exist in the system (by default, this happens during installation).

File permissions

As a general security rule, web server content must not belong to the process running the server. In our case, the files should not belong to the `apache` user and group, since it has write access to the folders.

You assign the contents to the unprivileged user or to the root user and the associated group. Incidentally, you also take the opportunity to restrict the group's access rights.

```
cd /var/www/html
sudo chown -R root:root ./*
sudo find ./ -type d -exec chmod 0755 "{}" \;
sudo find ./ -type f -exec chmod 0644 "{}" \;
```

5. Part 3. Application servers

5.1 PHP and PHP-FPM

In this chapter, you will learn about PHP and PHP-FPM.

PHP (**P**HP **H**ypertext **P**reprocessor) is a source scripting language specially designed for web application development. In 2024, PHP represented a little less than 80% of the web pages generated in the world. PHP is open-source and is the core of the most famous CMS (WordPress, Drupal, Joomla!, Magento, and others.).

PHP-FPM (**F**astCGI **P**rocess **M**anager) is integrated to PHP since its version 5.3.3. The FastCGI version of PHP brings additional functionalities.

Objectives: In this chapter, you will learn how to:

✔ install a PHP application server ✔ configure PHP-FPM pool ✔ optimize a PHP-FPM application server

🏁 **PHP**, **PHP-FPM**, **Application server**

Knowledge: ★ ★ ★ **Complexity**: ★ ★ ★

Reading time: 30 minutes

5.1.1 Generalities

CGI (**C**ommon **G**ateway **I**nterface) and **FastCGI** allow communication between the web server (Apache or Nginx) and a development language (PHP, Python, Java):

- In the case of **CGI**, each request creates a **new process**, which is less efficient in performance.
- **FastCGI** relies on a **certain number of processes** to treat its client requests.

PHP-FPM, **in addition to better performances**, brings:

- The possibility of better **partitioning the applications**: launching processes with different uid/gid, with personalized `php.ini` files,
- The management of the statistics,
- Log management,
- Dynamic management of processes and restart without service interruption ('graceful').

> **Note**
>
> Since Apache has a PHP module, php-fpm is more commonly used on an Nginx server.

5.1.2 Choose a PHP version

Rocky Linux, like its upstream, offers many versions of the language. Some of them have reached the end of their life but are kept to continue hosting historical applications that are not yet compatible with new versions of PHP. Please refer to the supported versions page of the php.net website to choose a supported version.

To obtain a list of available versions, enter the following command:

9.3 PHP module list

```
$ sudo dnf module list php

Rocky Linux 9 - AppStream
Name
Stream
Profiles
Summary
php                                                 8.1
[d]                                                 common [d], devel, minimal

Hint: [d]efault, [e]nabled, [x]disabled, [i]nstalled
```

The Remi repository offers more recent releases of PHP than the Appstream repository, including versions 8.2 and 8.3.

To install the Remi repository, run the following command:

```
sudo dnf install https://rpms.remirepo.net/enterprise/remi-release-9.rpm
```

Enable the Remi repository by running the following command:

```
sudo dnf config-manager --set-enabled remi
```

You can now activate a newer module (PHP 8.3) by entering the following command:

```
sudo dnf module enable php:remi-8.3
```

8.9 PHP module list

```
$ sudo dnf module list php

Rocky Linux 8 - AppStream
Name
Stream
Profiles
Summary
php                                                 7.2
[d]                                                common [d], devel,
minimal                                                PHP scripting language
php                                                 7.
3                                                   common [d], devel,
minimal                                                PHP scripting language
php                                                 7.
4                                                   common [d], devel,
minimal                                                PHP scripting language
php                                                 8.
0                                                   common [d], devel,
minimal                                                PHP scripting language

Hint: [d]efault, [e]nabled, [x]disabled, [i]nstalled
```

Rocky provides different PHP modules from its AppStream repository.

You will note that the default version of a Rocky 8.9 is 7.2 that has already reached its end of life at the time of writing.

You can activate a newer module by entering the following command:

```
sudo dnf module enable php:8.0
================================================================================
Package               Architecture         Version
Repository             Size
================================================================================
Enabling module streams:
httpd                                      2.4
nginx                                      1.14
php                                        8.0

Transaction Summary
================================================================================

Is this ok [y/N]:

Transaction Summary
================================================================================

Is this ok [y/N]: y
Complete!
```

You can now proceed to the installation of the PHP engine.

5.1.3 Installation of the PHP cgi mode

First, install and use PHP in CGI mode. You can only make it work with the Apache web server and its `mod_php` module. You will see in the FastCGI part (php-fpm) of this document, how to integrate PHP in Nginx (but also Apache).

The installation of PHP is relatively trivial since it consists of installing the main package and the few modules you will need.

The example below installs PHP with the modules usually installed with it.

9.3 install PHP

```
sudo dnf install php php-cli php-gd php-curl php-zip php-mbstring
```

You will be prompted to import GPG keys for the epel9 (Extra Packages for Enterprise Linux 9) and Remi repositories during installation. Enter y to import the keys:

```
Extra Packages for Enterprise Linux 9 - x86_64
Importing GPG key 0x3228467C:
Userid     : "Fedora (epel9) <epel@fedoraproject.org>"
Fingerprint: FF8A D134 4597 106E CE81 3B91 8A38 72BF 3228 467C
From       : /etc/pki/rpm-gpg/RPM-GPG-KEY-EPEL-9
Is this ok [y/N]: y
Key imported successfully
Remi's RPM repository for Enterprise Linux 9 - x86_64
Importing GPG key 0x478F8947:
Userid     : "Remi's RPM repository (https://rpms.remirepo.net/)
<remi@remirepo.net>"
Fingerprint: B1AB F71E 14C9 D748 97E1 98A8 B195 27F1 478F 8947
From       : /etc/pki/rpm-gpg/RPM-GPG-KEY-remi.el9
Is this ok [y/N]: y
Key imported successfully
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.

Complete!
```

8.9 install PHP

```
sudo dnf install php php-cli php-gd php-curl php-zip php-mbstring
```

You can check that the installed version corresponds to the expected one:

9.3 check PHP version

```
$ php -v
PHP 8.3.2 (cli) (built: Jan 16 2024 13:46:41) (NTS gcc x86_64)
Copyright (c) The PHP Group
Zend Engine v4.3.2, Copyright (c) Zend Technologies
with Zend OPcache v8.3.2, Copyright (c), by Zend Technologies
```

8.9 check PHP version

```
$ php -v
PHP 7.4.19 (cli) (built: May  4 2021 11:06:37) ( NTS )
Copyright (c) The PHP Group
Zend Engine v3.4.0, Copyright (c) Zend Technologies
with Zend OPcache v7.4.19, Copyright (c), by Zend Technologies
```

5.1.4 Apache Integration

To serve PHP pages in CGI mode, you must install the Apache server, configure it, activate it, and start it.

- Installation:

```
sudo dnf install httpd
```

```
activation:
```

```
sudo systemctl enable --now httpd
sudo systemctl status httpd
```

- Do not forget to configure the firewall:

```
sudo firewall-cmd --add-service=http --permanent
sudo firewall-cmd --reload
```

The default vhost should work out of the box. PHP provides a `phpinfo()` function that generates a summary table of its configuration. It is useful to test the good working of PHP. However, be careful not to leave such test files on your servers. They represent a huge security risk for your infrastructure.

Create the file `/var/www/html/info.php` (`/var/www/html` being the default vhost directory of the default Apache configuration):

```
<?php
phpinfo();
?>
```

Use a web browser to check that the server works properly by going to the page http://your-server-ip/info.php.

> **Warning**
>
> Do not leave the `info.php` file on your server!

5.1.5 Installation of the PHP cgi mode (PHP-FPM)

Noted earlier, many advantages exist for switching web hosting to PHP-FPM mode.

The installation entails only the php-fpm package:

```
sudo dnf install php-fpm
```

As php-fpm is a service from a system point of view, you must activate and start it:

```
sudo systemctl enable --now php-fpm
sudo systemctl status php-fpm
```

Configuration of the PHP cgi mode

The main configuration file is `/etc/php-fpm.conf`.

```
include=/etc/php-fpm.d/*.conf
[global]
pid = /run/php-fpm/php-fpm.pid
error_log = /var/log/php-fpm/error.log
daemonize = yes
```

> **Note**
>
> The php-fpm configuration files are widely commented on. Go and have a look!

As you can see, the files in the `/etc/php-fpm.d/` directory with the `.conf` extension are always included.

By default, a PHP process pool declaration named `www`, is in `/etc/php-fpm.d/www.conf`.

```
[www]
user = apache
group = apache

listen = /run/php-fpm/www.sock
listen.acl_users = apache,nginx
listen.allowed_clients = 127.0.0.1

pm = dynamic
pm.max_children = 50
pm.start_servers = 5
pm.min_spare_servers = 5
pm.max_spare_servers = 35

slowlog = /var/log/php-fpm/www-slow.log

php_admin_value[error_log] = /var/log/php-fpm/www-error.log
php_admin_flag[log_errors] = on
php_value[session.save_handler] = files
php_value[session.save_path]    = /var/lib/php/session
php_value[soap.wsdl_cache_dir]  = /var/lib/php/wsdlcache
```

Instructions	Description
`[pool]`	Process pool name. The configuration file can comprise several process pools (the pool's name in brackets starts a new section).
`listen`	Defines the listening interface or the Unix socket used.

Configuring the way to access php-fpm processes

Two ways exist for connecting.

With an `inet-interface` such as:

`listen = 127.0.0.1:9000`.

Or with a UNIX socket:

`listen = /run/php-fpm/www.sock`.

Note

Using a socket when the web server and PHP server are on the same machine removes the TCP/IP layer and optimizes the performance.

When working with an interface, you have to configure `listen.owner`, `listen.group`, `listen.mode` to specify the owner, the owner group, and the rights of the UNIX socket. **Warning:** Both servers (web and PHP) must have access rights on the socket.

When working with a socket, you must configure `listen.allowed_clients` to restrict access to the PHP server to certain IP addresses.

Example: `listen.allowed_clients = 127.0.0.1`

Static or dynamic configuration

You can manage PHP-FPM processes statically or dynamically.

In static mode, `pm.max_children` sets a limit to the number of child processes:

```
pm = static
pm.max_children = 10
```

This configuration starts 10 processes.

In dynamic mode, PHP-FPM starts at *most* the number of processes specified by the value of `pm.max_children`. It first starts some processes corresponding to `pm.start_servers`, keeping at least the value of `pm.min_spare_servers` of inactive processes and at most `pm.max_spare_servers` of inactive processes.

Example:

```
pm = dynamic
pm.max_children = 5
pm.start_servers = 2
pm.min_spare_servers = 1
pm.max_spare_servers = 3
```

PHP-FPM will create a new process to replace one that has processed several requests equivalent to `pm.max_requests`.

By default the value of `pm.max_requests` is 0, meaning processes are never recycled. Using the `pm.max_requests` option can be interesting for applications with memory leaks.

A third mode of operation is the `ondemand` mode. This mode only starts a process when it receives a request. It is not an optimal mode for sites with strong influences and is reserved for specific needs (sites with very weak requests, management backend, and so on.).

> **Note**
>
> The configuration of the operating mode of PHP-FPM is essential to ensure the optimal functioning of your web server.

Process status

PHP-FPM offers, like Apache and its `mod_status` module, a page indicating the status of the process.

To activate the page, set its access path with the `pm.status_path` directive:

```
pm.status_path = /status
```

```
$ curl http://localhost/status_php
pool:                 www
process manager:      dynamic
start time:           03/Dec/2021:14:00:00 +0100
start since:          600
accepted conn:        548
listen queue:         0
max listen queue:     15
listen queue len:     128
idle processes:       3
active processes:     3
total processes:      5
max active processes: 5
max children reached: 0
slow requests:        0
```

Logging long requests

The `slowlog` directive specifies the file that receives logging requests that are too long (for instance, whose time exceeds the value of the `request_slowlog_timeout` directive).

The default location of the generated file is `/var/log/php-fpm/www-slow.log`.

```
request_slowlog_timeout = 5
slowlog = /var/log/php-fpm/www-slow.log
```

A value of 0 for `request_slowlog_timeout` disables logging.

5.1.6 NGinx integration

The default setting of nginx already includes the necessary configuration to make PHP work with PHP-FPM.

The configuration file `fastcgi.conf` (or `fastcgi_params`) is under `/etc/nginx/`:

```
fastcgi_param  SCRIPT_FILENAME    $document_root$fastcgi_script_name;
fastcgi_param  QUERY_STRING       $query_string;
fastcgi_param  REQUEST_METHOD     $request_method;
fastcgi_param  CONTENT_TYPE       $content_type;
fastcgi_param  CONTENT_LENGTH     $content_length;

fastcgi_param  SCRIPT_NAME        $fastcgi_script_name;
fastcgi_param  REQUEST_URI        $request_uri;
fastcgi_param  DOCUMENT_URI       $document_uri;
fastcgi_param  DOCUMENT_ROOT      $document_root;
fastcgi_param  SERVER_PROTOCOL    $server_protocol;
fastcgi_param  REQUEST_SCHEME     $scheme;
fastcgi_param  HTTPS              $https if_not_empty;

fastcgi_param  GATEWAY_INTERFACE  CGI/1.1;
fastcgi_param  SERVER_SOFTWARE    nginx/$nginx_version;

fastcgi_param  REMOTE_ADDR        $remote_addr;
fastcgi_param  REMOTE_PORT        $remote_port;
fastcgi_param  SERVER_ADDR        $server_addr;
fastcgi_param  SERVER_PORT        $server_port;
fastcgi_param  SERVER_NAME        $server_name;

# PHP only, required if PHP was built with --enable-force-cgi-redirect
fastcgi_param  REDIRECT_STATUS    200;
```

For nginx to process `.php` files, add the following directives to the site configuration file:

If PHP-FPM is listening on port 9000:

```
location ~ \.php$ {
  include /etc/nginx/fastcgi_params;
  fastcgi_pass 127.0.0.1:9000;
}
```

If php-fpm is listening on a UNIX socket:

```
location ~ \.php$ {
  include /etc/nginx/fastcgi_params;
  fastcgi_pass unix:/run/php-fpm/www.sock;
}
```

5.1.7 Apache integration

The configuration of Apache to use a PHP pool is quite simple. You have to use the proxy modules with a `ProxyPassMatch` directive, for example:

```
<VirtualHost *:80>
  ServerName web.rockylinux.org
  DocumentRoot "/var/www/html/current/public"

  <Directory "/var/www/html/current/public">
    AllowOverride All
    Options -Indexes +FollowSymLinks
    Require all granted
  </Directory>
  ProxyPassMatch ^/(.*\.php(/.*)?)$ "fcgi://127.0.0.1:9000/var/www/html/
current/public"

</VirtualHost>
```

5.1.8 Solid configuration of PHP pools

Optimizing the number of requests served and analyzing the memory used by the PHP scripts, is necessary to optimize the maximum amount of launched threads.

First of all, you need to know the average amount of memory used by a PHP process with the command:

```
while true; do ps --no-headers -o "rss,cmd" -C php-fpm | grep "pool www" | awk
'{ sum+=$1 } END { printf ("%d%s\n", sum/NR/1024,"Mb") }' >> avg_php_proc;
sleep 60; done
```

This will give you a pretty accurate idea of the average memory footprint of a PHP process on this server.

The result of the rest of this document is a memory footprint of 120MB per process at full load.

On a server with 8Gb of RAM, keeping 1Gb for the system and 1Gb for the OPCache (see the rest of this document), is 6Gb left to process PHP requests from clients.

You can conclude that this server can accept at most **50 threads** `((6*1024) / 120)`.

A good configuration of `php-fpm` specific to this use case is:

```
pm = dynamic
pm.max_children = 50
pm.start_servers = 12
pm.min_spare_servers = 12
pm.max_spare_servers = 36
pm.max_requests = 500
```

with:

- `pm.start_servers` = 25% of `max_children`
- `pm.min_spare_servers` = 25% of `max_children`
- `pm.max_spare_servers` = 75% of `max_children`

5.1.9 Opcache configuration

The `opcache` (Optimizer Plus Cache) is the first level of cache that you can influence.

It keeps the compiled PHP scripts in memory, which strongly impacts the execution of the web pages (removes the reading of the script on disk + the compilation time).

To configure it, you must work on:

- The size of the memory dedicated to the opcache according to the hit ratio, configuring it correctly
- The number of PHP scripts to cache (number of keys + maximum number of scripts)
- The number of strings to cache

To install it:

```
sudo dnf install php-opcache
```

To configure it, edit the `/etc/php.d/10-opcache.ini` configuration file:

```
opcache.memory_consumption=128
opcache.interned_strings_buffer=8
opcache.max_accelerated_files=4000
```

Where:

- `opcache.memory_consumption` corresponds to the amount of memory needed for the opcache (increase this until obtaining a correct hit ratio).
- `opcache.interned_strings_buffer` the amount of strings to cache.
- `opcache.max_accelerated_files` is near to the result of the `find ./ -iname "*.php"|wc -l` command.

You can refer to an `info.php` page (including the `phpinfo();`) to configure the opcache (see for example the values of `Cached scripts` and `Cached strings`).

Note

At each new deployment of new code, it will be necessary to empty the opcache (for example by restarting the php-fpm process).

Note

Do not underestimate the speed gain that can be achieved by setting up and configuring the opcache correctly.

6. Part 4. Databases servers

MySQL, MariaDB and PostgreSQL are open-source RDBMS (Relational DataBase Management System).

6.1 MariaDB and MySQL

In this chapter, you will learn about the RDBMS MariaDB and MySQL.

Objectives: In this chapter, you will learn how to:

✓ install, configure, and secure MariaDB server and MySQL server; ✓ perform some administrative actions on databases and users.

🏁 **RDBMS**, **database**, **MariaDB**, **MySQL**

Knowledge: ★ ★ ★ **Complexity**: ★ ★ ★

Reading time: 30 minutes

6.1.1 Generalities

MySQL was developed by Michael "Monty" Widenius (a Finnish computer scientist) who founded MySQL AB in 1995. MySQL AB was acquired by SUN in 2008, which in turn was acquired by Oracle in 2009, which still owns the MySQL software and distributes it under a dual GPL and proprietary license.

In 2009, Michael Widenius left SUN, founded Monty Program AB and launched the development of his community fork of MySQL : MariaDB under GPL license. Governance of the project is entrusted to the MariaDB Foundation, which ensures that the project remains free.

It was not long before the majority of Linux distributions offered MariaDB packages instead of MySQL ones, and major accounts such as Wikipedia and Google also adopted the community fork.

MySQL and MariaDB are among the world's most widely used RDBMSs (professionally and by the general public), particularly for web applications (**LAMP**: Linux + Apache + Mysql-MariaDB + Php).

Mysql-MariaDB's main competitors are:

- PostgreSQL,
- OracleDB,
- Microsoft SQL Server.

Databases services are multi-threaded and multi-user, run on most operating systems (Linux, Unix, BSD, Mac OSx, Windows), and are accessible from many programming languages (Php, Java, Python, C, C++, Perl, others).

Support is offered for several engines, enabling the assignment of different engines to different tables within the same database, depending on requirements:

MyISAM

: the simplest, but does not support transactions or foreign keys. It is an indexed sequential engine. MyISAM is now deprecated.

InnoDB

: manages table integrity (foreign keys and transactions), but takes up more disk space. This has been the default engine since MySQL version 5.6. It is a transactional engine.

Memory

: tables are stored in memory.

Archive

: data compression on insertion saves disk space, but slows down search queries (cold data).

It is a matter of adopting an engine according to need: Archive for log storage, Memory for temporary data, and so on.

MariaDB/MySQL uses port 3306/tcp for network communication.

As the default version supplied with Rocky is the MariaDB community version of the database, this chapter will deal with this version. Only the differences between MySQL and MariaDB are specifically dealt with.

6.1.2 Installation

Use the `dnf` command to install the `mariadb-server` package:

```
sudo dnf install -y mariadb-server
```

By default, the version installed on a Rocky 9 is 10.5.

Activate the service at startup and start it:

```
sudo systemctl enable mariadb --now
```

You can check the status of the `mariadb` service:

```
sudo systemctl status mariadb
```

To install a more recent version, you'll need to use the `dnf` modules:

```
$ sudo dnf module list mariadb
Last metadata expiration check: 0:00:09 ago on Thu Jun 20 11:39:10 2024.
Rocky Linux 9 - AppStream
Name                          Stream
Profiles                                      Summary
mariadb                       10.11                       client, galera,
server [d]                      MariaDB Module

Hint: [d]efault, [e]nabled, [x]disabled, [i]nstalled
```

If you have not yet installed the mariadb server, activating the desired module version will suffice:

```
$ sudo dnf module enable mariadb:10.11
Last metadata expiration check: 0:02:23 ago on Thu Jun 20 11:39:10 2024.
Dependencies resolved.
================================================================================
```

```
Package                          Architecture
Version                             Repository                          Size
================================================================================
Enabling module streams:
 mariadb                                                             10.11

Transaction Summary
================================================================================
Is this ok [y/N]: y
Complete!
```

You can now install the package. The desired version will be automatically installed:

```
sudo dnf install -y mariadb-server
```

About default users

Please note the logs provided by mariadb at first start (`/var/log/messages`):

```
mariadb-prepare-db-dir[6560]: Initializing MariaDB database
mariadb-prepare-db-dir[6599]: Two all-privilege accounts were created.
mariadb-prepare-db-dir[6599]: One is root@localhost, it has no password, but
you need to
mariadb-prepare-db-dir[6599]: be system 'root' user to connect. Use, for
example, sudo mysql
mariadb-prepare-db-dir[6599]: The second is mysql@localhost, it has no password
either, but
mariadb-prepare-db-dir[6599]: you need to be the system 'mysql' user to
connect.
mariadb-prepare-db-dir[6599]: After connecting you can set the password, if you
would need to be
mariadb-prepare-db-dir[6599]: able to connect as any of these users with a
password and without sudo
```

6.1.3 Configuration

Configuration files can are in `/etc/my.cnf` and `/etc/my.cnf.d/` .

Some important default options have been setup in the `/etc/my.cnf.d/mariadb-server.cnf` :

```
[server]

[mysqld]
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock
log-error=/var/log/mariadb/mariadb.log
pid-file=/run/mariadb/mariadb.pid
...
```

As you can see, data is in the `/var/lib/mysql` per default. This folder can require a lot of storage space and recurring volume increases. It is therefore advisable to mount this folder on a dedicated partition.

6.1.4 Security

MariaDB and Mysql include a script to help you secure your server. It remove for example remote root logins and sample users, the less-secure default options.

Use the `mariadb-secure-installation` and secure your server:

```
sudo mariadb-secure-installation
```

The script will prompt you to provide a password for your root user.

Note

The `mysql_secure_installation` command is now a symlink to the `mariadb-secure-installation` command:

```
$ ll /usr/bin/mysql_secure_installation
lrwxrwxrwx. 1 root root 27 Oct 12  2023 /usr/bin/mysql_secure_installation -> mariadb-secure-installation
```

If providing a password each time you have to use mariadb's commands is a problem, you can set up a `~/.my.cnf` file with your credentials, that will be used per default by mariadb to connect to your server.

```
[client]
user="root"
password="#######"
```

Ensure the permissions are restrictive enough to only allow the current user can access:

```
chmod 600 ~/.my.cnf
```

Warning

This is not the best way. There is another solution more secure than storing a password in plain text. Since MySQL 5.6.6, it is now possible to store your credentials in an encrypted login `.mylogin.cnf`, thanks to the `mysql_config_editor` command.

If your server runs a firewall (which is a good thing), you might need to consider opening it, but only if you need your service accessible from the outside.

```
sudo firewall-cmd --zone=public --add-service=mysql
sudo firewall-cmd --reload
```

Note

The best security is not to open your database server to the outside world (if the application server is hosted on the same server), or to restrict access to authorized IPs only.

6.1.5 Administration

The `mariadb` command

The `mariadb` command is a simple SQL shell that supports interactive and non-interactive use.

```
mysql -u user -p [base]
```

Option	Information
`-u user`	Provides a username to connect with.
`-p`	Asks for a password.
`base`	The database to connect to.

Note

The `mysql` command is now a symlink to the `mariadb` command:

```
$ ll /usr/bin/mysql
lrwxrwxrwx. 1 root root 7 Oct 12  2023 /usr/bin/mysql -> mariadb
```

Example:

```
$ sudo mariadb -u root
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 15
Server version: 10.5.22-MariaDB MariaDB Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
+--------------------+
3 rows in set (0.003 sec)
```

The `mariadb-admin` command

The `mariadb-admin` command is a client for administering a MariaDB server.

```
mariadb-admin -u user -p command
```

Option	Information
`-u user`	Provides a username to connect with.
`-p`	Asks for a password.
`command`	A command to execute.

The `mariadb-admin` provides several commands as `version`, `variables`, `stop-slave` or `start-slaves`, `create databasename`, and so on.

Example:

```
mariadb-admin -u root -p version
```

Note

The `mysqladmin` command is now a symlink to the `mariadb-admin` command:

```
$ ll /usr/bin/mysqladmin
lrwxrwxrwx. 1 root root 13 Oct 12  2023 /usr/bin/mysqladmin -> mariadb-admin
```

6.1.6 About logs

MariaDB provides various logs:

- **Error log**: This contains messages generated at service startup and shutdown, as well as important events (warnings and errors).
- **Binary log**: This log (in binary format) records all actions that modify database structure or data. If you need to restore a database, you will need to restore the backup AND replay the binary log to recover the state of the database before the crash.
- **Query log**: All client requests are logged here.
- **Slow requests log**: Slow queries, i.e. those that take longer than a set time to execute, are logged separately in this log. By analyzing this file, you may be able to take steps to reduce execution time (e.g., by setting up indexes or modifying the client application).

With the exception of the binary log, these logs are in text format, so they can be used directly!

To enable logging of long requests, edit the `my.cnf` configuration file to add the following lines:

```
slow_query_log      = 1
slow_query_log_file = /var/log/mysql/mysql-slow.log
long_query_time     = 2
```

The minimum value for the `long_query_time` variable is 0 and the default value is `10` seconds.

Restart the service for the changes to take effect.

Once the log file is full, you can analyze it with the `mariadb-dumpslow` command.

```
mariadb-dumpslow [options] [log_file ...]
```

Option	Information
`-t n`	Displays only the first n queries.
`-s sort_type`	Sorts by number of queries.
`-r`	Inverts results display.

Sort types can be :

Option	Information
`c`	according to number of requests.
`c`	according to number of requests.
`t` or `at`	according to execution time or average execution time (a for average).
`l` or `al`	according to lock time or its average.
`r` or `aR`	as a function of the number of lines returned or its average.

6.1.7 About backup

As with any RDBMS, backing up a database is done while the data modification is off-line. You can do this by:

- stopping the service, known as an offline backup;
- while the service is running, buy temporarily locking out updates (suspending all modifications). This is an online backup.
- using a snapshot of the LVM file system, enabling the backing up of data with a cold file system.

The backup format can be an ASCII (text) file, representing the state of the database and its data in the form of SQL commands, or a binary file, corresponding to MySQL storage files.

While you can back up a binary file using common utilities such as tar or cpio, an ASCII file requires a utility such as `mariadb-dump`.

The `mariadb-dump` command can perform a dump of your database.

During the process, locking of some data access occurs.

```
mariadb-dump -u root -p DATABASE_NAME > backup.sql
```

> **Note**
>
> Do not forget that after restoring a full backup, restoring the binary files (binlogs) completes the reconstitution of the data.

The resulting file is usable to restore the database data. The database must still exist or you must have recreated it beforehand!:

```
mariadb -u root -p DATABASE_NAME < backup.sql
```

6.1.8 Graphical tools

Graphical tools exist to facilitate the administration and management of database data. Here are a few examples:

- DBeaver

6.1.9 Workshop

In this workshop, you will install, configure, and secure your mariadb server.

Task 1 : Installation

Install the mariadb-server package:

```
$ sudo dnf install mariadb-server
Last metadata expiration check: 0:10:05 ago on Thu Jun 20 11:26:03 2024.
Dependencies resolved.
================================================================================
Package                                    Architecture
Version                          Repository                  Size
================================================================================
Installing:
 mariadb-server                             x86_64                 3:
10.5.22-1.el9_2                  appstream                 9.6 M
Installing dependencies:
...
```

Installation adds a `mysql` user to the system, with `/var/lib/mysql` as home directory:

```
$ cat /etc/passwd
...
mysql:x:27:27:MySQL Server:/var/lib/mysql:/sbin/nologin
...
```

Enable and start the service:

```
$ sudo systemctl enable mariadb --now
Created symlink /etc/systemd/system/mysql.service → /usr/lib/systemd/system/
mariadb.service.
Created symlink /etc/systemd/system/mysqld.service → /usr/lib/systemd/system/
mariadb.service.
Created symlink /etc/systemd/system/multi-user.target.wants/mariadb.service → /
usr/lib/systemd/system/mariadb.service.
```

Check the installation:

```
$ sudo systemctl status mariadb
● mariadb.service - MariaDB 10.5 database server
     Loaded: loaded (/usr/lib/systemd/system/mariadb.service; enabled; preset:
disabled)
     Active: active (running) since Thu 2024-06-20 11:48:56 CEST; 1min 27s ago
       Docs: man:mariadbd(8)
             https://mariadb.com/kb/en/library/systemd/
    Process: 6538 ExecStartPre=/usr/libexec/mariadb-check-socket (code=exited,
status=0/SUCCESS)
    Process: 6560 ExecStartPre=/usr/libexec/mariadb-prepare-db-dir
mariadb.service (code=exited, status=0/SUCCESS)
    Process: 6658 ExecStartPost=/usr/libexec/mariadb-check-upgrade
(code=exited, status=0/SUCCESS)
   Main PID: 6643 (mariadbd)
     Status: "Taking your SQL requests now..."
      Tasks: 9 (limit: 11110)
     Memory: 79.5M
        CPU: 1.606s
     CGroup: /system.slice/mariadb.service
             └─6643 /usr/libexec/mariadbd --basedir=/usr

Jun 20 11:48:56 localhost.localdomain mariadb-prepare-db-dir[6599]: The second
is mysql@localhost, it has no password either, but
Jun 20 11:48:56 localhost.localdomain mariadb-prepare-db-dir[6599]: you need
to be the system 'mysql' user to connect.
```

```
Jun 20 11:48:56 localhost.localdomain mariadb-prepare-db-dir[6599]: After
connecting you can set the password, if you would need to be
Jun 20 11:48:56 localhost.localdomain mariadb-prepare-db-dir[6599]: able to
connect as any of these users with a password and without sudo
Jun 20 11:48:56 localhost.localdomain mariadb-prepare-db-dir[6599]: See the
MariaDB Knowledgebase at https://mariadb.com/kb
Jun 20 11:48:56 localhost.localdomain mariadb-prepare-db-dir[6599]: Please
report any problems at https://mariadb.org/jira
Jun 20 11:48:56 localhost.localdomain mariadb-prepare-db-dir[6599]: The latest
information about MariaDB is available at https://mariadb.org>Jun 20 11:48:56
localhost.localdomain mariadb-prepare-db-dir[6599]: Consider joining MariaDB's
strong and vibrant community:
Jun 20 11:48:56 localhost.localdomain mariadb-prepare-db-dir[6599]: https://
mariadb.org/get-involved/
Jun 20 11:48:56 localhost.localdomain systemd[1]: Started MariaDB 10.5
database server.
```

Try connecting to the server:

```
$ sudo mariadb
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 9
Server version: 10.5.22-MariaDB MariaDB Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
+--------------------+
3 rows in set (0.001 sec)

MariaDB [(none)]> exit
Bye
```

```
$ sudo mariadb-admin version
mysqladmin  Ver 9.1 Distrib 10.5.22-MariaDB, for Linux on x86_64
Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Server version          10.5.22-MariaDB
```

```
Protocol version        10
Connection              Localhost via UNIX socket
UNIX socket             /var/lib/mysql/mysql.sock
Uptime:                 7 min 24 sec

Threads: 1  Questions: 9  Slow queries: 0  Opens: 17  Open tables: 10  Queries
per second avg: 0.020
```

As you can see, the `root` user does not need to provide a password. You will correct that during the next task.

Task 2 : Secure your server

Launch the `mariadb-secure-installation` and follow the instructions:

```
$ sudo mariadb-secure-installation

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, we'll need the current
password for the root user. If you've just installed MariaDB, and
haven't set the root password yet, you should just press enter here.

Enter current password for root (enter for none):
OK, successfully used password, moving on...

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

You already have your root account protected, so you can safely answer 'n'.

Switch to unix_socket authentication [Y/n] y
Enabled successfully!
Reloading privilege tables..
 ... Success!


You already have your root account protected, so you can safely answer 'n'.

Change the root password? [Y/n] y
New password:
Re-enter new password:
Password updated successfully!
Reloading privilege tables..
 ... Success!
```

```
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] y
 ... 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] y
 ... 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] y
 - 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] y
 ... 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!
```

Try connecting again, with and without password to your server:

```
$ mariadb -u root
ERROR 1045 (28000): Access denied for user 'root'@'localhost' (using password:
NO)

$ mariadb -u root -p
```

```
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 4
Server version: 10.5.22-MariaDB MariaDB Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>
```

Configure your firewall:

```
sudo firewall-cmd --zone=public --add-service=mysql --permanent
sudo firewall-cmd --reload
```

Task 3 : Testing the installation

Verify your installation :

```
$ mysqladmin -u root -p version
Enter password:
mysqladmin  Ver 9.1 Distrib 10.5.22-MariaDB, for Linux on x86_64
Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Server version          10.5.22-MariaDB
Protocol version        10
Connection              Localhost via UNIX socket
UNIX socket             /var/lib/mysql/mysql.sock
Uptime:                 29 min 18 sec

Threads: 1  Questions: 35  Slow queries: 0  Opens: 20  Open tables: 13
Queries per second avg: 0.019
```

The `version` give you information about the server.

Task 4 : Create a new database and a user

Create a new database:

```
MariaDB [(none)]> create database NEW_DATABASE_NAME;
```

Create a new user and give him all rights on all table of that database:

```
MariaDB [(none)]> grant all privileges on NEW_DATABASE_NAME.* TO
'NEW_USER_NAME'@'localhost' identified by 'PASSWORD';
```

Replace `localhost` per `%` if you want to grant access from everywhere or replace per IP addresses if you can.

You can restrict the priveleges granted. There are different types of permissions to offer users:

- **SELECT**: read data
- **USAGE**: authorization to connect to the server (given by default when a new user is created)
- **INSERT**: add new tuples to a table.
- **UPDATE**: modify existing tuples
- **DELETE**: delete tuples
- **CREATE**: create new tables or databases
- **DROP**: delete existing tables or databases
- **ALL PRIVILEGES**: all rights
- **GRANT OPTION**: give or remove rights to other users

Do not forget to reload apply the new rights:

```
MariaDB [(none)]> flush privileges;
```

Check:

```
$ mariadb -u NEW_USER_NAME -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 8
Server version: 10.5.22-MariaDB MariaDB Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
```

```
MariaDB [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| NEW_DATABASE_NAME  |
| information_schema |
+--------------------+
2 rows in set (0.001 sec)
```

Add sample data into your database:

```
$ mariadb -u NEW_USER_NAME -p NEW_DATABASE_NAME
MariaDB [NEW_DATABASE_NAME]> CREATE TABLE users(
    id INT NOT NULL AUTO_INCREMENT,
    first_name VARCHAR(30) NOT NULL,
    last_name VARCHAR(30) NOT NULL,
    age INT DEFAULT NULL,
    PRIMARY KEY (id));
Query OK, 0 rows affected (0.017 sec)

MariaDB [NEW_DATABASE_NAME]> INSERT INTO users (first_name, last_name, age)
VALUES ("Antoine", "Le Morvan", 44);
Query OK, 1 row affected (0.004 sec)
```

Task 5 : Create a remote user

In this task, you will create a new user, grant access from remote, and test a connection with that user.

```
MariaDB [(none)]> grant all privileges on NEW_DATABASE_NAME.* TO
'NEW_USER_NAME'@'%' identified by 'PASSWORD';
Query OK, 0 rows affected (0.005 sec)

MariaDB [(none)]> flush privileges;
Query OK, 0 rows affected (0.004 sec)
```

Use this user and the `-h` option to connect remotely to your server:

```
$ mariadb -h YOUR_SERVER_IP -u NEW_USER_NAME -p NEW_DATABASE_NAME
Enter password:
...

MariaDB [NEW_DATABASE_NAME]>
```

Task 6 : Perform an upgrade

Enable the module needed:

```
$ sudo dnf module enable mariadb:10.11
[sudo] password for antoine:
Last metadata expiration check: 2:00:16 ago on Thu Jun 20 11:50:27 2024.
Dependencies resolved.
================================================================================
Package                          Architecture
Version                             Repository                           Size
================================================================================
module streams:
 mariadb                                                              10.11

Transaction Summary
================================================================================
Is this ok [y/N]: y
Complete!
```

Upgrade the packages:

```
$ sudo dnf update mariadb
Last metadata expiration check: 2:00:28 ago on Thu Jun 20 11:50:27 2024.
Dependencies resolved.
================================================================================
Package                            Architecture
Version
Repository                Size
================================================================================
Upgrading:
 mariadb                             x86_64               3:
10.11.6-1.module+el9.4.0+20012+a68bdff7               appstream               1.
7 M
 mariadb-backup                      x86_64               3:
10.11.6-1.module+el9.4.0+20012+a68bdff7               appstream               6.
7 M
 mariadb-common                      x86_64               3:
10.11.6-1.module+el9.4.0+20012+a68bdff7               appstream
28 k
 mariadb-errmsg                      x86_64               3:
10.11.6-1.module+el9.4.0+20012+a68bdff7               appstream
254 k
 mariadb-gssapi-server               x86_64               3:
10.11.6-1.module+el9.4.0+20012+a68bdff7               appstream
15 k
```

```
 mariadb-server                     x86_64              3:
10.11.6-1.module+el9.4.0+20012+a68bdff7               appstream
10 M
 mariadb-server-utils               x86_64              3:
10.11.6-1.module+el9.4.0+20012+a68bdff7               appstream
261 k

Transaction Summary
================================================================================
Upgrade  7 Packages

Total download size: 19 M
Is this ok [y/N]: y
Downloading Packages:
(1/7): mariadb-gssapi-
server-10.11.6-1.module+el9.4.0+20012+a68bdff7.x86_64.rpm
99 kB/s |  15 kB     00:00
(2/7): mariadb-server-
utils-10.11.6-1.module+el9.4.0+20012+a68bdff7.x86_64.rpm
1.1 MB/s | 261 kB     00:00
(3/7): mariadb-
errmsg-10.11.6-1.module+el9.4.0+20012+a68bdff7.x86_64.rpm
2.5 MB/s | 254 kB     00:00
(4/7): mariadb-
common-10.11.6-1.module+el9.4.0+20012+a68bdff7.x86_64.rpm
797 kB/s |  28 kB     00:00
(5/7):
mariadb-10.11.6-1.module+el9.4.0+20012+a68bdff7.x86_64.rpm
5.7 MB/s | 1.7 MB     00:00
(6/7): mariadb-
server-10.11.6-1.module+el9.4.0+20012+a68bdff7.x86_64.rpm
9.5 MB/s |  10 MB     00:01
(7/7): mariadb-
backup-10.11.6-1.module+el9.4.0+20012+a68bdff7.x86_64.rpm
7.7 MB/s | 6.7 MB     00:00
--------------------------------------------------------------------------------
Total
13 MB/s |  19 MB     00:01
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction

...

Complete!
```

Your databases now need upgrading (check your `/var/log/messages` as the service complains):

```
mariadb-check-upgrade[8832]: The datadir located at /var/lib/mysql needs to be
upgraded using 'mariadb-upgrade' tool. This can be done using the following
steps:
mariadb-check-upgrade[8832]:  1. Back-up your data before with 'mariadb-
upgrade'
mariadb-check-upgrade[8832]:  2. Start the database daemon using 'systemctl
start mariadb.service'
mariadb-check-upgrade[8832]:  3. Run 'mariadb-upgrade' with a database user
that has sufficient privileges
mariadb-check-upgrade[8832]: Read more about 'mariadb-upgrade' usage at:
mariadb-check-upgrade[8832]: https://mariadb.com/kb/en/mysql_upgrade/
```

Do not forget to execute the upgrade script provided by MariaDB:

```
sudo mariadb-upgrade
Major version upgrade detected from 10.5.22-MariaDB to 10.11.6-MariaDB. Check
required!
Phase 1/8: Checking and upgrading mysql database
Processing databases
mysql
mysql.column_stats                                 OK
mysql.columns_priv                                 OK
mysql.db                                           OK
...
Phase 2/8: Installing used storage engines... Skipped
Phase 3/8: Running 'mysql_fix_privilege_tables'
Phase 4/8: Fixing views
mysql.user                                         OK
...
Phase 5/8: Fixing table and database names
Phase 6/8: Checking and upgrading tables
Processing databases
NEW_DATABASE_NAME
information_schema
performance_schema
sys
sys.sys_config                                     OK
Phase 7/8: uninstalling plugins
Phase 8/8: Running 'FLUSH PRIVILEGES'
OK
```

Task 6 : Perform a dump

The `mariadb-dump` command can perform a dump of your database.

```
mariadb-dump -u root -p NEW_DATABASE_NAME > backup.sql
```

Verify:

```
cat backup.sql
-- MariaDB dump 10.19  Distrib 10.11.6-MariaDB, for Linux (x86_64)
--
-- Host: localhost    Database: NEW_DATABASE_NAME
-- ------------------------------------------------------
-- Server version       10.11.6-MariaDB

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
...

--
-- Table structure for table `users`
--

DROP TABLE IF EXISTS `users`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `users` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `first_name` varchar(30) NOT NULL,
  `last_name` varchar(30) NOT NULL,
  `age` int(11) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=latin1
COLLATE=latin1_swedish_ci;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `users`
--

LOCK TABLES `users` WRITE;
/*!40000 ALTER TABLE `users` DISABLE KEYS */;
INSERT INTO `users` VALUES
(1,'Antoine','Le Morvan',44);
/*!40000 ALTER TABLE `users` ENABLE KEYS */;
UNLOCK TABLES;
/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;
```

```
...
-- Dump completed on 2024-06-20 14:32:41
```

6.1.10 Check your Knowledge

✔ Which database version installs by default?

MySQL 5.5

MariaDB 10.5

MariaDB 11.11

Mysql 8

✔ Which command do you use to apply rights changes?

flush rights

flush privileges

mariadb reload

apply

6.1.11 Conclusion

In this chapter, you have installed and secured a MariaDB database server, created a database and a dedicated user.

These skills are a prerequisite for the administration of your databases.

In the next section, you will see how to install the MySQL database instead of the MariaDB fork.

6.2 Mysql

In this chapter, you will learn how to install MySQL server.

Only notable differences between the MariaDB and MySQL versions are included.

Objectives: In this chapter, you will learn how to:

✔ install, configure and secure MariaDB server and MySQL server;

🏁 **RDBMS**, **database**, **MariaDB**, **MySQL**

Knowledge: ⭐ ⭐ ⭐ **Complexity**: ⭐ ⭐ ⭐

Reading time: 10 minutes

6.2.1 Installation of MySQL

By default, the installed version of MySQL is version 8.0.

This time, you have to install the `mysql-server` package:

```
sudo dnf install mysql-server
```

and start the `mysqld` service:

```
sudo systemctl enable mysqld.service --now
```

You can now follow the previous chapter replacing the following commands:

- `mariadb` => `mysql`
- `mariadb-admin` => `mysql_admin`
- `mariadb-dump` => `mysql_dump`
- `mariadb-secure-installation` => `mysql_secure_installation`

To install the latest version of mysql-server, you will have to install a different repository.

Visit this page: https://dev.mysql.com/downloads/repo/yum/ and copy the repository URL.

For example:

```
sudo dnf install -y https://dev.mysql.com/get/mysql84-community-release-
el9-1.noarch.rpm
```

When completed, you can perform the `dnf update`:

```
$ dnf update
Error: This command has to be run with superuser privileges (under the root
user on most systems).
[antoine@localhost ~]$ sudo dnf update
MySQL 8.4 LTS Community
Server
377 kB/s | 226 kB     00:00
MySQL Connectors
Community
110 kB/s |  53 kB     00:00
MySQL Tools 8.4 LTS
Community
170 kB/s |  97 kB     00:00
Dependencies resolved.
================================================================================
Package                                  Architecture
Version                                Repository                           Size
================================================================================
 mysql-community-client                    x86_64            8.
4.0-1.el9                              mysql-8.4-lts-community            3.1 M
     replacing  mysql.x86_64 8.0.36-1.el9_3
 mysql-community-server                    x86_64            8.
4.0-1.el9                              mysql-8.4-lts-community             50 M
     replacing  mariadb-connector-c-config.noarch 3.2.6-1.el9_0
     replacing  mysql-server.x86_64 8.0.36-1.el9_3
Installing dependencies:
  ...

Transaction Summary
================================================================================
7 Packages

Total download size: 59 M
Is this ok [y/N]: y
Downloading Packages:
(1/7): mysql-community-client-
plugins-8.4.0-1.el9.x86_64.rpm
3.4 MB/s | 1.4 MB     00:00
(2/7): mysql-community-
common-8.4.0-1.el9.x86_64.rpm
1.3 MB/s | 576 kB     00:00
(3/7): mysql-community-icu-data-
```

```
files-8.4.0-1.el9.x86_64.rpm
30 MB/s | 2.3 MB     00:00
(4/7): mysql-community-
client-8.4.0-1.el9.x86_64.rpm
5.8 MB/s | 3.1 MB     00:00
(5/7): mysql-community-
libs-8.4.0-1.el9.x86_64.rpm
6.8 MB/s | 1.5 MB     00:00
(6/7): net-
tools-2.0-0.62.20160912git.el9.x86_64.rpm
1.1 MB/s | 292 kB     00:00
(7/7): mysql-community-
server-8.4.0-1.el9.x86_64.rpm
48 MB/s |  50 MB     00:01
--------------------------------------------------------------------------------
Total
30 MB/s |  59 MB     00:01
MySQL 8.4 LTS Community
Server
3.0 MB/s | 3.1 kB     00:00
Importing GPG key 0xA8D3785C:
 Userid     : "MySQL Release Engineering <mysql-build@oss.oracle.com>"
 Fingerprint: BCA4 3417 C3B4 85DD 128E C6D4 B7B3 B788 A8D3 785C
 From       : /etc/pki/rpm-gpg/RPM-GPG-KEY-mysql-2023
Is this ok [y/N]: y
Key imported successfully
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :
  ...
Installed:
  mysql-community-server-8.4.0-1.el9.x86_64
  ...

Complete!
```

Do not forget to re-enable and restart your server:

```
sudo systemctl enable mysqld.service --now
```

6.2.2 Check your Knowledge MySQL

✔ Which MySQL database version is installed by default?

MySQL 5.5

MariaDB 10.5

MariaDB 11.11

Mysql 8

6.3 Secondary server with MariaDB

In this chapter, you will learn how to configure a Primary/Secondary system servers with MariaDB.

Objectives: In this chapter, you will learn how to:

✔ activate the binlogs in your servers; ✔ setup a secondary server to replicate data from primary server.

🏁 **MariaDB**, **Replication**, **Primary**, **Secondary**

Knowledge: ★ ★ **Complexity**: ★ ★ ★

Reading time: 10 minutes

6.3.1 Generalities secondary server with MariaDB

As soon as you start using your database more intensively, you will need to replicate your data on several servers.

This can be done in several ways:

- Distribute write requests to the primary server and read requests to the secondary server.
- Perform database backups on the secondary server, which avoids blocking writes to the primary server for the duration of the backups.

If your usage becomes even more demanding, you may consider switching to a primary/primary system: replications are then made crosswise, but beware of the risk of blocking the uniqueness of primary keys. Otherwise, you will need to switch to a more advanced clustering system.

6.3.2 Configuration secondary server with MariaDB

How to activate the binlogs

Perform this action on the primary and secondary servers:

Add the following options to your `/etc/my.cnf.d/mariadb-server.cnf` file, under the `[mariadb]` key:

```
[mariadb]
log-bin
server_id=1
log-basename=server1
binlog-format=mixed
```

for the primary server, and for the secondary server:

```
[mariadb]
log-bin
server_id=2
log-basename=server2
binlog-format=mixed
```

The `server_id` option must be unique on each server in the cluster, while the `log-basename` option allows you to specify a prefix to the binlog files. If you do not do this, you will not be able to rename your server in the future.

You can now restart the mariadb service on both servers:

```
sudo systemctl restart mariadb
```

You can check that binlogs files are well created:

```
$ ll /var/lib/mysql/
total 123332
```

```
...
-rw-rw----. 1 mysql mysql         0 Jun 21 11:07 multi-master.info
drwx------. 2 mysql mysql      4096 Jun 21 11:07 mysql
srwxrwxrwx. 1 mysql mysql         0 Jun 21 11:16 mysql.sock
-rw-rw----. 1 mysql mysql       330 Jun 21 11:16 server1-bin.000001
-rw-rw----. 1 mysql mysql        21 Jun 21 11:16 server1-bin.index
...
```

How to configure the replication

First of all, on the primary, you will need to create users authorized to replicate data (be careful to restrict the IPs authorized):

```
$ sudo mariadb

MariaDB [(none)]> CREATE USER 'replication'@'%' IDENTIFIED BY 'PASSWORD';
Query OK, 0 rows affected (0.002 sec)

MariaDB [(none)]> GRANT REPLICATION SLAVE ON *.* TO 'replication'@'%';
Query OK, 0 rows affected (0.002 sec)
```

or better for security (change '192.168.1.101' with your own secondary IP):

```
$ sudo mariadb

MariaDB [(none)]> CREATE USER 'replication'@'192.168.1.101' IDENTIFIED BY
'PASSWORD';
Query OK, 0 rows affected (0.002 sec)

MariaDB [(none)]> GRANT REPLICATION SLAVE ON *.* TO
'replication'@'192.168.1.101';
Query OK, 0 rows affected (0.002 sec)
```

If your primary server already contains data, you will need to lock new transactions while the exporting or importing of data occurs to the secondary server(s), and tell the secondary servers when to start replication. If your server does not yet contain any data, the procedure is greatly simplified.

Prevent any changes to the data while you view the binary log position:

```
$ sudo mariadb

MariaDB [(none)]> FLUSH TABLES WITH READ LOCK;
```

```
Query OK, 0 rows affected (0.021 sec)

MariaDB [(none)]> SHOW MASTER STATUS;
+--------------------+----------+--------------+------------------+
| File               | Position | Binlog_Do_DB | Binlog_Ignore_DB |
+--------------------+----------+--------------+------------------+
| server1-bin.000001 |     1009 |              |                  |
+--------------------+----------+--------------+------------------+
1 row in set (0.000 sec)
```

Do not quit your session to keep the lock.

Record the File and Position details.

If your server contains data, it is time to create a backup and import it onto your secondary server(s). Keep the lock for the duration of the backup, and release it as soon as the backup is complete. This reduces downtime (the time it takes to copy and import the data on the secondary servers).

You can remove the lock now:

```
$ sudo mariadb

MariaDB [(none)]> UNLOCK TABLES;
Query OK, 0 rows affected (0.000 sec)
```

On the secondary server, you can now ready to setup the primary server to replicate with:

```
MariaDB [(none)]> CHANGE MASTER TO
  MASTER_HOST='192.168.1.100',
  MASTER_USER='replication',
  MASTER_PASSWORD='PASSWORD',
  MASTER_PORT=3306,
  MASTER_LOG_FILE='server1-bin.000001',
  MASTER_LOG_POS=1009,
  MASTER_CONNECT_RETRY=10;
Query OK, 0 rows affected, 1 warning (0.021 sec)

MariaDB [(none)]> START SLAVE;
Query OK, 0 rows affected (0.001 sec)
```

Replace the primary server IP with yours and the `MASTER_LOG_FILE` and `MASTER_LOG_POS` values with those you previously registered.

Check if the replication is ok:

```
MariaDB [(none)]> SHOW SLAVE STATUS \G
*************************** 1. row ***************************
               Slave_IO_State: Waiting for master to send event
                  Master_Host: 192.168.1.100
                  Master_User: replication
              Master_Log_File: server1-bin.000001
          Read_Master_Log_Pos: 1009
...
         Seconds_Behind_Master: 0
       Slave_SQL_Running_State: Slave has read all relay log; waiting for more
updates
...
1 row in set (0.001 sec)
```

The `Seconds_Behind_Master` is an interesting value to monitor as it can help you see if there is a replication issue.

6.3.3 Workshop secondary server using MariaDB

For this workshop, you will need two servers with MariaDB services installed, configured and secured as described in the previous chapters.

You will configure replication on the secondary server, then create a new database, insert data into it and check that the data is accessible on the secondary server.

Our two servers have the following IP addresses:

- server1: 192.168.1.100
- server2: 192.168.1.101

Remember to replace these values with your own.

Task 1: Create a dedicated replication user

On the primary server:

```
$ sudo mariadb

MariaDB [(none)]> CREATE USER 'replication'@'192.168.1.101' IDENTIFIED BY
'PASSWORD';
Query OK, 0 rows affected (0.002 sec)

MariaDB [(none)]> GRANT REPLICATION SLAVE ON *.* TO
'replication'@'192.168.1.101';
Query OK, 0 rows affected (0.002 sec)
```

Task 2: Record the primary server values

```
$ sudo mariadb

MariaDB [(none)]> FLUSH TABLES WITH READ LOCK;
Query OK, 0 rows affected (0.021 sec)

MariaDB [(none)]> SHOW MASTER STATUS;
+--------------------+----------+--------------+------------------+
| File               | Position | Binlog_Do_DB | Binlog_Ignore_DB |
+--------------------+----------+--------------+------------------+
| server1-bin.000001 |     1009 |              |                  |
+--------------------+----------+--------------+------------------+
1 row in set (0.000 sec)

MariaDB [(none)]> UNLOCK TABLES;
Query OK, 0 rows affected (0.000 sec)
```

Task 3: Activate the replication

On the secondary server:

```
MariaDB [(none)]> CHANGE MASTER TO
  MASTER_HOST='192.168.1.100',
  MASTER_USER='replication',
  MASTER_PASSWORD='PASSWORD',
  MASTER_PORT=3306,
  MASTER_LOG_FILE='server1-bin.000001',
  MASTER_LOG_POS=1009,
  MASTER_CONNECT_RETRY=10;
Query OK, 0 rows affected, 1 warning (0.021 sec)

MariaDB [(none)]> START SLAVE;
Query OK, 0 rows affected (0.001 sec)
```

Check if the replication is ok:

```
MariaDB [(none)]> SHOW SLAVE STATUS \G
*************************** 1. row ***************************
               Slave_IO_State: Waiting for master to send event
                  Master_Host: 192.168.1.100
                  Master_User: replication
              Master_Log_File: server1-bin.000001
          Read_Master_Log_Pos: 1009
...
         Seconds_Behind_Master: 0
       Slave_SQL_Running_State: Slave has read all relay log; waiting for more
updates
...
1 row in set (0.001 sec)
```

Task 4: Create a new database and a user

On the primary:

```
MariaDB [(none)]> create database NEW_DATABASE_NAME;
Query OK, 1 row affected (0.002 sec)

MariaDB [(none)]> grant all privileges on NEW_DATABASE_NAME.* TO
'NEW_USER_NAME'@'localhost' identified by 'PASSWORD';
Query OK, 0 rows affected (0.004 sec)
```

On the secondary, check for creation of the database:

```
MariaDB [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| NEW_DATABASE_NAME  |
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
```

Magic !

On the secondary, try connecting the new user created on the primary:

```
$ mariadb -u NEW_USER_NAME -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.

MariaDB [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| NEW_DATABASE_NAME  |
| information_schema |
+--------------------+
2 rows in set (0.000 sec)
```

Task 5: Insert new data

Insert new data on the primary server:

```
MariaDB [(none)]> use NEW_DATABASE_NAME
Database changed

MariaDB [(none)]>  CREATE TABLE users(
    ->     id INT NOT NULL AUTO_INCREMENT,
    ->     first_name VARCHAR(30) NOT NULL,
    ->     last_name VARCHAR(30) NOT NULL,
    ->     age INT DEFAULT NULL,
    ->     PRIMARY KEY (id));

MariaDB [NEW_DATABASE_NAME]> INSERT INTO users (first_name, last_name, age)
VALUES ("Antoine", "Le Morvan", 44);
Query OK, 1 row affected (0.004 sec)
```

On the secondary, check that data are replicated:

```
MariaDB [(none)]> use NEW_DATABASE_NAME
Database changed

MariaDB [NEW_DATABASE_NAME]> show tables;
+----------------------------+
| Tables_in_NEW_DATABASE_NAME |
+----------------------------+
| users                      |
+----------------------------+
1 row in set (0.000 sec)

MariaDB [NEW_DATABASE_NAME]> SELECT * FROM users;
```

```
+----+------------+-----------+------+
| id | first_name | last_name | age  |
+----+------------+-----------+------+
|  1 | Antoine    | Le Morvan |   44 |
+----+------------+-----------+------+
1 row in set (0.000 sec)
```

6.3.4 Check your Knowledge secondary server with MariaDB

✔ Each server must have the same id within a cluster?

True

False

✔ Binary logs must be enabled before replication is activated.?

True

False

It depends

6.3.5 Conclusion secondary server with MariaDB

As you can see, creating one or more secondary servers is a relatively easy action, but it does require service interruption on the main server.

It does, however, offer many advantages: high data availability, load balancing, and simplified backup.

It goes without saying that, in the event of a main server crash, promotion of one of the secondary servers to main server can occur.

7. Part 5. Load balancing, caching and proxyfication

8. Part 6. Mail servers

9. Part 7. High availability

https://docs.rockylinux.org/