

How to Install PHP and PHP-FPM on Rocky Linux 9

Learn how to install PHP and PHP-FPM on Rocky Linux 9 with our step-by-step guide. Includes configuration tips and troubleshooting for optimal performance.

Contents

01	Introduction	3
02	Prerequisites	3
03	Install PHP	3
04	Install PHP Extensions	5
05	Configure PHP-FPM	6
06	Test PHP and PHP-FPM	9
07	Install Multiple PHP Versions	11
08	Conclusion	15

Introduction

PHP (Hypertext Preprocessor) is an open-source server-side scripting language that enables the development of dynamic web applications. PHP-FPM (FastCGI Process Manager) is an implementation of PHP FastCGI that enables processing of PHP requests using pools configurations with optimized performance and faster response times.

This article explains how to install PHP and PHP-FPM on Rocky Linux 9, configure the default pool to enable integrations with other applications such as web servers and run sample dynamic web applications.

Prerequisites

Before you begin:

- Deploy a [Rocky Linux 9](#) instance on Vultr.
- Access the server using SSH as a non-root user with sudo privileges.
- [Update the server](#).

Install PHP

PHP is available in the default repositories on Rocky Linux 9 with the latest version information. Follow the steps below to install PHP and PHP-FPM using the default DNF package manager on your server.

1. Update the server packages index.

CONSOLE

```
$ sudo dnf update -y
```

2. Install PHP.

CONSOLE

```
$ sudo dnf install -y php
```

3. View the installed PHP on your server.

CONSOLE

```
$ php -v
```

Output:

```
PHP 8.0.30 (cli) (built: Aug  3 2023 17:13:08) ( NTS gcc x86_64 )
Copyright (c) The PHP Group
Zend Engine v4.0.30, Copyright (c) Zend Technologies
    with Zend OPcache v8.0.30, Copyright (c), by Zend Technologies
```

4. View the installed PHP-FPM version and verify that it matches your PHP version.

CONSOLE

```
$ php-fpm -v
```

Output:

```
PHP 8.0.30 (fpm-fcgi) (built: Aug  3 2023 17:13:08)
Copyright (c) The PHP Group
Zend Engine v4.0.30, Copyright (c) Zend Technologies
    with Zend OPcache v8.0.30, Copyright (c), by Zend Technologies
```

Install PHP Extensions

PHP extensions enable additional functionalities and integration of dynamic features required by modern web applications. Follow the steps below to install common PHP extensions required by most web applications.

1. Install common PHP extensions.

CONSOLE

```
$ sudo dnf install -y php-mysqlnd php-pgsql php-redis php-curl php-json php-gd php-xml php-mbstring php-zip
```

The above command installs the following PHP extensions:

- php-mysqlnd: Enables PHP to connect and interact with MySQL databases.
- php-pgsql: Enables PHP to interact with PostgreSQL databases.
- php-redis: Allows PHP scripts to interact with Redis databases.
- php-curl: Enables PHP applications to create requests to other servers or http services.
- php-json: Provides JSON handling functionalities.
- php-gd: Enables PHP to create and manipulate images.
- php-xml: Provides functionalities to parse and generate XML contents in PHP applications.
- php-mbstring: Enables PHP to process multi-byte encodings.
- php-zip: Enables PHP to read and modify ZIP archives.

2. View the installed PHP modules on your server.

CONSOLE

```
$ php -m
```

Output:

```
[PHP Modules]
.....
json
libxml
mbstring
msgpack
mysqli
mysqlnd
openssl
pcntl
pcre
PDO
pdo_mysql
pdo_pgsql
pdo_sqlite
pgsql
Phar
readline
redis
Reflection
session
SimpleXML
.....
[Zend Modules]
Zend OPcache
```

Configure PHP-FPM

PHP-FPM is available and included in the PHP package during installation. PHP-FPM acts as a gateway to the PHP processes and handles multiple PHP requests depending on your pool configurations. Follow the steps below to configure the default PHP-FPM pool to process requests depending on your server performance.

1. Switch to the PHP-FPM configuration files directory.

```
CONSOLE
```

```
$ cd /etc/php-fpm.d/
```

2. Open the default PHP-FPM pool configuration `www.conf`.

CONSOLE

```
$ sudo nano /etc/php-fpm.d/www.conf
```

- Verify the default pool name `www`.

INI

```
[www]
```

- Find the following user and group directives to enable PHP-FPM run with the same user as your web server. For example, change the default value `apache` to `nginx` when using the Nginx web server and vice versa.

INI

```
user = nginx  
group = nginx
```

- Find the following `listen` directive and verify the PHP-FPM UNIX socket path to use when connecting to the pool.

INI

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

- Find the `pm.max_children` directive and set the value depending on your available server memory. A higher value enables processing of multiple workers while a lower value may enforce queuing.

```
INI
```

```
pm.max_children = 50
```

- Find and uncomment the `pm.max_requests` directive to set the number of processes each child process can run before it's terminated.

```
INI
```

```
pm.max_requests = 500
```

Save and close the file.

3. Enable the PHP-FPM service to start automatically start at boot time.

```
CONSOLE
```

```
$ sudo systemctl enable php-fpm
```

4. Start the PHP-FPM service.

```
CONSOLE
```

```
$ sudo systemctl start php-fpm
```

5. View the PHP-FPM service status and verify that it's running.

```
CONSOLE
```

```
$ sudo systemctl status php-fpm
```

Output:

```
● php-fpm.service - The PHP FastCGI Process Manager
   Loaded: loaded (/usr/lib/systemd/system/php-fpm.service; enabled; preset:
disabled)
   Active: active (running) since Thu 2024-06-27 02:23:24 UTC; 7s ago
 Main PID: 51711 (php-fpm)
```

```
Status: "Ready to handle connections"  
Tasks: 6 (limit: 48896)  
Memory: 13.9M  
CPU: 42ms  
CGroup: /system.slice/php-fpm.service  
├─51711 "php-fpm: master process (/etc/php-fpm.conf)"  
├─51712 "php-fpm: pool www"  
├─51713 "php-fpm: pool www"  
├─51714 "php-fpm: pool www"  
├─51715 "php-fpm: pool www"  
└─51716 "php-fpm: pool www"
```

Test PHP and PHP-FPM

PHP includes a default web server package that lets you run applications by specifying a specific listen address. Follow the steps below to set up a sample PHP application and use the built-in PHP web server utility to run the application.

1. Create a new directory to store your PHP application files. For example, `/var/www/php-test`.

CONSOLE

```
$ sudo mkdir -p /var/www/php-test
```

2. Create a new sample PHP application file `info.php`.

CONSOLE

```
$ sudo nano /var/www/php-test/info.php
```

3. Add the following contents to the file.

PHP

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

Save and close the file.

The above application code displays full information about your PHP installation including the version, and installed modules on the server.

4. Run the application directory using the built-in PHP web server utility on the HTTP port `80`.

CONSOLE

```
$ php -S 0.0.0.0:80 -t /var/www/php-test &
```

Output:

```
PHP 8.0.30 Development Server (http://0.0.0.0:80) started
```

5. Allow connections to the HTTP port `80` through the default `firewalld` utility.

CONSOLE

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

6. Reload the firewall configuration to apply changes.

```
$ sudo firewall-cmd --reload
```

7. Access your server IP using a web browser such as Chrome and load the `/info.php` path to access your PHP application.

```
http://SERVER-IP/info.php
```

PHP Version 8.0.30	
System	Linux vcd-807-publish-pass 5.14.0-427.22.1.el9_4.x86_64 #1 SMP PREEMPT_DYNAMIC Wed Jun 19 17:35:04 UTC 2024 x86_64
Build Date	Aug 3 2023 17:13:08
Build System	Rocky Linux release 9.2 (Blue Onyx)
Build Provider	Rocky Enterprise Software Foundation
Compiler	gcc (GCC) 11.3.1 20221121 (Red Hat 11.3.1-4)
Architecture	x86_64
Server API	Built-in HTTP server
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc
Loaded Configuration File	/etc/php.ini
Scan this dir for additional .ini files	/etc/php.d

Install Multiple PHP Versions

Running multiple PHP projects requires specific package versions depending on your deployment environment. You can run multiple PHP versions on your server that include specific features or functionalities available in a specific PHP version required by your application. Follow the steps below to install the `phpbrew` utility and manage multiple PHP versions on your server.

1. Update your server's package index.

CONSOLE

```
$ sudo dnf update -y
```

2. Install the EPEL repository on your server.

CONSOLE

```
$ sudo dnf install epel-release -y
```

3. Enable the CRB (CodeReady Builder) repository.

CONSOLE

```
$ sudo dnf config-manager --set-enabled crb
```

4. Install all required dependencies packages.

CONSOLE

```
$ sudo dnf install -y gcc gcc-c++ kernel-devel make autoconf automake bzip2-devel libcurl-devel libjpeg-devel libpng-devel freetype-devel libxml2-devel libxslt-devel libzip-devel oniguruma-devel readline-devel libmcrypt-devel libpq-devel openldap-devel openssl-devel re2c bison
```

5. Download the latest `phpbrew` binary file to install on your server.

CONSOLE

```
$ curl -L -O https://github.com/phpbrew/phpbrew/raw/master/phpbrew
```

6. Enable execution privileges on the file.

CONSOLE

```
$ sudo chmod +x phpbrew
```

7. Move the binary file to the `/usr/local/bin` directory to enable `phpbrew` as a system-wide command.

CONSOLE

```
$ sudo mv phpbrew /usr/local/bin/phpbrew
```

8. View the installed `phpbrew` version to verify access to the utility.

CONSOLE

```
$ phpbrew -v
```

Output:

```
____ _ _ _ _  
| _ _ \ | | | | _ _ \ _ _ \
```

```
| | / / | | | | / / | / / _ _ _ _ _ _ _ _  
| _ / | _ | | _ / | _ \ ' _ / _ \ \ \ / /  
| | | | | | | | | | / / | | _ \ \ \ / /  
\ | \ | | \ \ | \ _ / | | \ | \ \ /
```

Brew your latest php!

.....

phpbrew 2.1.0

powered by <https://github.com/c9s/CLIFramework>

9. Initialize `phpbrew` to work in your user environment.

CONSOLE

```
$ phpbrew init
```

Output:

```
Using root: /home/exampleuser/.phpbrew  
Initialization successfully finished!  
<=====>  
Phpbrew environment is initialized, required directories are created under  
  
/home/exampleuser/.phpbrew
```

10. Run the following command to add your local `phpbrew` directory to the `.bashrc` file and enable your configurations to work in multiple terminal sessions. Replace `exampleuser` with your actual user.

CONSOLE

```
$ echo "[[ -e /home/exampleuser/.phpbrew/bashrc ]] &&  
source /home/exampleuser/.phpbrew/bashrc" >> /home/  
exampleuser/.bashrc
```

11. Reload the `.bashrc` file to apply changes.

CONSOLE

```
$ source /home/exampleuser/.bashrc
```

12. Install one or more PHP versions on your server. For example:

- Install PHP 8.1 with the `+default` and `+mysql` extensions.

CONSOLE

```
$ phpbrew install php-8.1 +default +mysql
```

- Install PHP 8.2 with the `+default` and `+mysql` extensions.

CONSOLE

```
$ phpbrew install php-8.2 +default +mysql
```

- Install PHP 8.3 with the `+default` and `+mysql` extensions.

CONSOLE

```
$ phpbrew install php-8.3 +default +mysql
```

13. View all installed versions on your server.

CONSOLE

```
$ phpbrew list
```

Output:

```
* (system)
php-8.1.29
php-8.2.20
php-8.3.8
```

14. View the active PHP version on your server.

CONSOLE

```
$ php -v
```

Output:

```
PHP 8.0.30 (cli) (built: Aug  3 2023 17:13:08) ( NTS gcc x86_64 )
Copyright (c) The PHP Group
Zend Engine v4.0.30, Copyright (c) Zend Technologies
    with Zend OPcache v8.0.30, Copyright (c), by Zend Technologies
```

15. Switch to a specific PHP version available on your `phpbrew` list.

CONSOLE

```
$ phpbrew switch php-8.3.8
```

16. View the active PHP version in your user environment.

CONSOLE

```
$ php -v
```

Output:

```
PHP 8.3.8 (cli) (built: Jun 27 2024 03:04:29) (NTS)
Copyright (c) The PHP Group
Zend Engine v4.3.8, Copyright (c) Zend Technologies
```

Based on the above output, the PHP version `8.3.8` is active in your user's environment. When using multiple PHP versions on your server, ensure to create independent PHP-FPM pools to define specific resource limits for each PHP version depending on your project requirements.

Conclusion

You have installed PHP and PHP-FPM on your Rocky Linux 9 server to efficiently process dynamic web applications. In addition, you installed common PHP extensions to enable additional application functionalities and switched

between multiple versions to match your project needs. For more information and PHP-FPM configuration options, visit the official [PHP documentation](#).



VULTR

