Release Notes

Versioning Scheme

Laravel and its other first-party packages follow Semantic Versioning. Major framework releases are released every year (~Q1), while minor and patch releases may be released as often as every week. Minor and patch releases should **never** contain breaking changes.

When referencing the Laravel framework or its components from your application or package, you should always use a version constraint such as `^12.0`, since major releases of Laravel do include breaking changes. However, we strive to always ensure you may update to a new major release in one day or less.

Named Arguments

Named arguments are not covered by Laravel's backwards compatibility guidelines. We may choose to rename function arguments when necessary in order to improve the Laravel codebase. Therefore, using named arguments when calling Laravel methods should be done cautiously and with the understanding that the parameter names may change in the future.

Support Policy

For all Laravel releases, bug fixes are provided for 18 months and security fixes are provided for 2 years. For all additional libraries, including Lumen, only the latest major release receives bug fixes. In addition, please review the database versions supported by Laravel.

Version	PHP (*)	Release	Bug Fixes Until	Security Fixes Until
9	8.0 - 8.2	February 8th, 2022	August 8th, 2023	February 6th, 2024
10	8.1 - 8.3	February 14th, 2023	August 6th, 2024	February 4th, 2025
11	8.2 - 8.4	March 12th, 2024	September 3rd, 2025	March 12th, 2026
12	8.2 - 8.4	February 24th, 2025	August 13th, 2026	February 24th, 2027

End of life
Security fixes only

(*) Supported PHP versions

Laravel 12

Laravel 12 continues the improvements made in Laravel 11.x by updating upstream dependencies and introducing new starter kits for React, Vue, and Livewire, including the option of using WorkOS AuthKit for user authentication. The WorkOS variant of our starter kits offers social authentication, passkeys, and SSO support.

Minimal Breaking Changes

Much of our focus during this release cycle has been minimizing breaking changes. Instead, we have dedicated ourselves to shipping continuous quality-of-life improvements throughout the year that do not break existing applications.

Therefore, the Laravel 12 release is a relatively minor "maintenance release" in order to upgrade existing dependencies. In light of this, most Laravel applications may upgrade to Laravel 12 without changing any application code.

New Application Starter Kits

Laravel 12 introduces new application starter kits for React, Vue, and Livewire. The React and Vue starter kits utilize Inertia 2, TypeScript, shadcn/ui, and Tailwind, while the Livewire starter kits utilize the Tailwind-based Flux UI component library and Laravel Volt.

The React, Vue, and Livewire starter kits all utilize Laravel's built-in authentication system to offer login, registration, password reset, email verification, and more. In addition, we are introducing a WorkOS AuthKit-powered variant of each starter kit, offering social authentication, passkeys, and SSO support. WorkOS offers free authentication for applications up to 1 million monthly active users.

With the introduction of our new application starter kits, Laravel Breeze and Laravel Jetstream will no longer receive additional updates.

To get started with our new starter kits, check out the starter kit documentation.

Upgrade Guide

High Impact Changes

Medium Impact Changes

Low Impact Changes

Upgrading To 12.0 From 11.x

Estimated Upgrade Time: 5 Minutes

> [!NOTE] We attempt to document every possible breaking change. Since some of these breaking changes are in obscure parts of the framework only a portion of these changes may actually affect your application. Want to save time? You can use Laravel Shift to help automate your application upgrades.

Updating Dependencies

Likelihood Of Impact: High

You should update the following dependencies in your application's `composer.json` file:

- `laravel/framework` to `^12.0`
- `phpunit/phpunit` to `^11.0`
- `pestphp/pest` to `^3.0`

Carbon 3

Likelihood Of Impact: Low

Support for Carbon 2.x has been removed. All Laravel 12 applications now require Carbon 3.x.

Updating the Laravel Installer

If you are using the Laravel installer CLI tool to create new Laravel applications, you should update your installer installation to be compatible with Laravel 12.x and the new Laravel starter kits. If you installed the Laravel installer via `composer global require`, you may update the installer using `composer global update`:

```
composer global update laravel/installer
```

If you originally installed PHP and Laravel via `php.new`, you may simply re-run the `php.new` installation commands for your operating system to install the latest version of PHP and the Laravel installer:

```
/bin/bash -c "$(curl -fsSL https://php.new/install/mac/8.4)"
```

```
# Run as administrator...
Set-ExecutionPolicy Bypass -Scope Process -Force;
[System.Net.ServicePointManager]::SecurityProtocol =
[System.Net.ServicePointManager]::SecurityProtocol -bor 3072; iex ((New-Object
System.Net.WebClient).DownloadString('https://php.new/install/windows/8.4'))
```

```
/bin/bash -c "$(curl -fsSL https://php.new/install/linux/8.4)"
```

Or, if you are using Laravel Herd's bundled copy of the Laravel installer, you should update your Herd installation to the latest release.

Authentication

Updated `DatabaseTokenRepository` Constructor Signature

Likelihood Of Impact: Very Low

The constructor of the `Illuminate\Auth\Passwords\DatabaseTokenRepository` class now expects the `$expires` parameter to be given in seconds, rather than minutes.

Concurrency

Concurrency Result Index Mapping

Likelihood Of Impact: Low

When invoking the `Concurrency::run` method with an associative array, the results of the concurrent operations are now returned with their associated keys:

```
$result = Concurrency::run([
    'task-1' => fn () => 1 + 1,
    'task-2' => fn () => 2 + 2,
]);

// ['task-1' => 2, 'task-2' => 4]
```

Container

Container Class Dependency Resolution

Likelihood Of Impact: Low

The dependency injection container now respects the default value of class properties when resolving a class instance. If you were previously relying on the container to resolve a class instance without the default value, you may need to adjust your application to account for this new behavior:

```
class Example
{
    public function __construct(public ?Carbon $date = null) {}
}

$example = resolve(Example::class);

// <= 11.x
$example->date instanceof Carbon;

// >= 12.x
$example->date === null;
```

Database

Multi-Schema Database Inspecting

Likelihood Of Impact: Low

The `Schema::getTables()`, `Schema::getViews()`, and `Schema::getTypes()` methods now include the results from all schemas by default. You may pass the `schema` argument to retrieve the result for the given schema only:

```
// All tables on all schemas...
$tables = Schema::getTables();

// All tables on the 'main' schema...
$table = Schema::getTables(schema: 'main');

// All tables on the 'main' and 'blog' schemas...
$table = Schema::getTables(schema: ['main', 'blog']);
```

The `Schema::getTableListing()` method now returns schema-qualified table names by default. You may pass the `schemaQualified` argument to change the behavior as desired:

```
$tables = Schema::getTableListing();
// ['main.migrations', 'main.users', 'blog.posts']

$table = Schema::getTableListing(schema: 'main');
// ['main.migrations', 'main.users']

$table = Schema::getTableListing(schema: 'main', schemaQualified: false);
// ['migrations', 'users']
```

The `db:table` and `db:show` commands now output the results of all schemas on MySQL, MariaDB, and SQLite, just like PostgreSQL and SQL Server.

Eloquent

Models and UUIDv7

Likelihood Of Impact: Medium

The `HasUuids` trait now returns UUIDs that are compatible with version 7 of the UUID spec (ordered UUIDs). If you would like to continue using ordered UUIDv4 strings for your model's IDs, you should now use the `HasVersion4Uuids` trait:

```
use Illuminate\Database\Eloquent\Concerns\HasUuids; // [tl! remove]
use Illuminate\Database\Eloquent\Concerns\HasVersion4Uuids as HasUuids; // [tl!
add]
```

The `HasVersion7Uuids` trait has been removed. If you were previously using this trait, you should use the `HasUuids` trait instead, which now provides the same behavior.

Requests

Nested Array Request Merging

Likelihood Of Impact: Low

The `$request->mergeIfMissing()` method now allows merging nested array data using "dot" notation. If you were previously relying on this method to create a top-level array key containing the "dot" notation version of the key, you may need to adjust your application to account for this new behavior:

```
$request->mergeIfMissing([
    'user.last_name' => 'Otwell',
]);
```

Validation

Image Validation Now Excludes SVGs

The `image` validation rule no longer allows SVG images by default. If you would like to allow SVGs when using the `image` rule, you must explicitly allow them:

```
use Illuminate\Validation\Rules\File;

'photo' => 'required|image:allow_svg'

// Or...
'photo' => ['required', File::image(allowSvg: true)],
```

Miscellaneous

We also encourage you to view the changes in the `laravel/laravel` GitHub repository. While many of these changes are not required, you may wish to keep these files in sync with your application. Some of these changes will be covered in this upgrade guide, but others, such as changes to configuration files or comments, will not be. You can easily view the changes with the GitHub comparison tool and choose which updates are important to you.

Contribution Guide

Bug Reports

To encourage active collaboration, Laravel strongly encourages pull requests, not just bug reports. Pull requests will only be reviewed when marked as "ready for review" (not in the "draft" state) and all tests for new features are passing. Lingering, non-active pull requests left in the "draft" state will be closed after a few days.

However, if you file a bug report, your issue should contain a title and a clear description of the issue. You should also include as much relevant information as possible and a code sample that demonstrates the issue. The goal of a bug report is to make it easy for yourself - and others - to replicate the bug and develop a fix.

Remember, bug reports are created in the hope that others with the same problem will be able to collaborate with you on solving it. Do not expect that the bug report will automatically see any activity or that others will jump to fix it. Creating a bug report serves to help yourself and others start on the path of fixing the problem. If you want to chip in, you can help out by fixing any bugs listed in our issue trackers. You must be authenticated with GitHub to view all of Laravel's issues.

If you notice improper DocBlock, PHPStan, or IDE warnings while using Laravel, do not create a GitHub issue. Instead, please submit a pull request to fix the problem.

The Laravel source code is managed on GitHub, and there are repositories for each of the Laravel projects:

- Laravel Application
- Laravel Art
- Laravel Documentation
- Laravel Dusk
- Laravel Cashier Stripe
- Laravel Cashier Paddle
- Laravel Echo
- Laravel Envoy
- Laravel Folio
- Laravel Framework
- Laravel Homestead (Build Scripts)
- Laravel Horizon
- Laravel Livewire Starter Kit
- Laravel Passport
- Laravel Pennant
- Laravel Pint
- Laravel Prompts
- Laravel React Starter Kit
- Laravel Reverb
- Laravel Sail
- Laravel Sanctum
- Laravel Scout
- Laravel Socialite
- Laravel Telescope
- Laravel Vue Starter Kit
- Laravel Website

Support Questions

Laravel's GitHub issue trackers are not intended to provide Laravel help or support. Instead, use one of the following channels:

- GitHub Discussions
- Laracasts Forums
- Laravel.io Forums
- StackOverflow
- Discord
- Larachat
- IRC

Core Development Discussion

You may propose new features or improvements of existing Laravel behavior in the Laravel framework repository's GitHub discussion board. If you propose a new feature, please be willing to implement at least some of the code that would be needed to complete the feature.

Informal discussion regarding bugs, new features, and implementation of existing features takes place in the `#internals` channel of the Laravel Discord server. Taylor Otwell, the maintainer of Laravel, is typically present in the channel on weekdays from 8am-5pm (UTC-06:00 or America/Chicago), and sporadically present in the channel at other times.

Which Branch?

All bug fixes should be sent to the latest version that supports bug fixes (currently `12.x`). Bug fixes should **never** be sent to the `master` branch unless they fix features that exist only in the upcoming release.

Minor features that are **fully backward compatible** with the current release may be sent to the latest stable branch (currently `12.x`).

Major new features or features with breaking changes should always be sent to the `master` branch, which contains the upcoming release.

Compiled Assets

If you are submitting a change that will affect a compiled file, such as most of the files in `resources/css` or `resources/js` of the `laravel/laravel` repository, do not commit the compiled files. Due to their large size, they cannot realistically be reviewed by a maintainer. This could be exploited as a way to inject malicious code into Laravel. In order to defensively prevent this, all compiled files will be generated and committed by Laravel maintainers.

Security Vulnerabilities

If you discover a security vulnerability within Laravel, please send an email to Taylor Otwell at taylor@laravel.com. All security vulnerabilities will be promptly addressed.

Coding Style

Laravel follows the PSR-2 coding standard and the PSR-4 autoloading standard.

PHPDoc

Below is an example of a valid Laravel documentation block. Note that the `@param` attribute is followed by two spaces, the argument type, two more spaces, and finally the variable name:

```
/**
 * Register a binding with the container.
 *
 * @param  string|array  $abstract
 * @param  \Closure|string|null  $concrete
 * @param  bool  $shared
 * @return void
 *
 * @throws \Exception
 */
public function bind($abstract, $concrete = null, $shared = false)
{
    // ...
}
```

When the `@param` or `@return` attributes are redundant due to the use of native types, they can be removed:

```
/**
 * Execute the job.
 */
public function handle(AudioProcessor $processor): void
{
    //
}
```

However, when the native type is generic, please specify the generic type through the use of the `@param` or `@return` attributes:

```
/**
 * Get the attachments for the message.
 * @return array<int, \Illuminate\Mail\Mailables\Attachment>
 */
public function attachments(): array
{
    return [
        Attachment::fromStorage('/path/to/file'),
    ];
}
```

StyleCI

Don't worry if your code styling isn't perfect! StyleCI will automatically merge any style fixes into the Laravel repository after pull requests are merged. This allows us to focus on the content of the contribution and not the code style.

Code of Conduct

The Laravel code of conduct is derived from the Ruby code of conduct. Any violations of the code of conduct may be reported to Taylor Otwell (taylor@laravel.com):

- Participants will be tolerant of opposing views.
- Participants must ensure that their language and actions are free of personal attacks and disparaging personal remarks.
- When interpreting the words and actions of others, participants should always assume good intentions.
- Behavior that can be reasonably considered harassment will not be tolerated.