Laravel 6.3 to 7.30 Upgrade Documentation

This document outlines the detailed steps required to upgrade a Laravel project from version 6.3 to 7.30. The upgrade includes updates to the composer.json file, adjustments to key package versions, and modifications to Handler.php to ensure compatibility with Laravel 7.30.

Approximate time required for this upgrade: 10-15 hours

Why Laravel 7.30 and Not Laravel 7.*?

We have specifically chosen Laravel 7.30 instead of using a wildcard version like 7.* because Laravel 7.30 is the latest and most stable release in the Laravel 7 series. It includes all relevant bug fixes, security updates, and performance improvements up to the final release in that version line.

Locking the version to 7.30 ensures consistent behavior and avoids unexpected issues that might arise from future patch-level changes. It also guarantees compatibility with the project's dependencies, such as altek/accountant, arcanedev/log-viewer, and others.

Although Laravel 7 is not an LTS (Long-Term Support) release, using the final stable version within the series provides a more predictable and secure environment, especially for production deployments. This approach minimizes upgrade risks and ensures a stable foundation for the application.

---

1. Backup the Project and Database

Backup the code: **cp -r laravel-adminpanel laravel-adminpanel-backup**

Or use Git:

```
git add .
git commit -m "Backup before Laravel 7 upgrade"
git tag pre-laravel-7
git push origin dev-upgrade-6-to-7 --tags
```

Backup the database:

For MySQL: **mysqldump -u youruser -p yourdatabase > db_backup.sql**

---

2. Update Laravel Version in composer.json

Edit composer.json: "laravel/framework": "^7.30"
Also, verify compatibility of any Laravel-specific packages with Laravel 7.

---

3. Try to Update Composer Packages

Run: **composer update**
If you encounter memory errors: COMPOSER_MEMORY_LIMIT=-1 composer update

---

4. Fix Package Compatibility Issues

Problem 1: Package X requires laravel/framework ^6.0, found ^7.30
Solution: composer require vendor/package:^X.Y

Problem 2: Abandoned packages or those not compatible with Laravel 7.
Solutions:

- Check the GitHub repo for Laravel 7 support
- Replace it with an alternative package
- Temporarily remove it from composer.json

Then retry: **composer update**

---

5. Clear and Optimize Laravel Setup

After a successful update, run the following commands to clear and optimize the Laravel setup:

php artisan config:clear
php artisan route:clear
php artisan view:clear
php artisan cache:clear
composer dump-autoload
php artisan optimize

---

6. Handle Code Changes (e.g., Handler.php)

Laravel 7 introduces changes to core files like app/Exceptions/Handler.php.

- Adjust method signatures

- Remove deprecated methods
- Add missing imports

---

7. Check Project Functionality

Start the server: **php artisan serve**
Verify:

- All pages load correctly
- Forms and submissions work
- Admin panel, login/logout, dashboard
- All JS and CSS assets load properly
- No console or network errors

---

8. Run Migrations and Test Data

Run migrations: **php artisan migrate**
If needed: **php artisan db:seed**

---

9. Run Tests (If Available)

php artisan test

---

10. Final Cleanup

php artisan config:cache
php artisan route:cache
php artisan view:cache

If using GrumPHP: vendor/bin/php-cs-fixer fix
Commit without GrumPHP check: git commit **--no-verify** -m "Laravel upgraded to 7.30"

Summary of Commands

- Backup
  cp -r laravel-adminpanel laravel-adminpanel-backup
  mysqldump -u root -p dbname > backup.sql

- Update

composer update
COMPOSER_MEMORY_LIMIT=-1 composer update

- Fix packages
  composer require vendor/package:^version

- Clear + optimize
  php artisan config:clear
  php artisan optimize
  php artisan migrate
  php artisan serve
  php artisan test

- Commit
  git commit -am "Upgraded Laravel to 7.30"
  git push

Composer Package Changes:

1. Updated Packages:

PHP Version:

Old Version: ^7.3
New Version: ^7.4

Reason for Change: During the upgrade to Laravel 7.30, we encountered compatibility issues with PHP 7.3. Laravel 7.x officially supports PHP 7.4 and higher. PHP 7.3 is no longer maintained, and PHP 7.4 brings enhanced performance, better error handling, and additional features such as typed properties. To ensure compatibility and take advantage of the improvements in PHP 7.4, we updated the PHP version in the composer.json

altek/accountant:

Old Version: ^1.2
New Version: ^2.1

Reason for Change: We upgraded to version 2.1 of altek/accountant to ensure compatibility with Laravel 7.30. The previous version (1.2) had issues working with newer versions of Laravel. Version 2.1 provides bug fixes, improved functionality, and better support for the dependencies used in Laravel 7.x.

arcanedev/log-viewer:

Old Version: ^5.0
New Version: ^7.0

Reason for Change: We encountered compatibility issues with the older version of arcanedev/log-viewer (5.0) when upgrading to Laravel 7.x. Version 7.0 resolves these issues, offering improved performance, bug fixes, and compatibility with Laravel 7.x, which was necessary for proper logging and debugging functionality.

laravel/framework:

Old Version: ^6.3
New Version: ^7.30

Reason for Change: While upgrading to Laravel 7.30, we faced compatibility issues with the older Laravel 6.x version. Upgrading to Laravel 7.30 resolved these issues and provided new features like improved route caching and job batching. The upgrade was essential to ensure smooth application performance and compatibility with the Laravel ecosystem.

nunomaduro/collision:

Old Version: ^3.0
New Version: ^4.0

Reason for Change: After upgrading to Laravel 7.30, the older version of nunomaduro/collision (3.0) caused errors due to incompatibility with Laravel 7.x. Upgrading to version 4.0 resolved these errors, providing better error handling, UI improvements, and necessary bug fixes for a smoother developer experience with the new Laravel version.

2. Added Packages:

laravel/ui:

New Version: ^2.0

Reason for Addition: In Laravel 7.x, the authentication scaffolding that was previously bundled by default is now separated into the laravel/ui package. This package provides the necessary frontend components for authentication (like login, registration, and password resets) that were previously included in Laravel core. Adding laravel/ui ensures that we can still use authentication features in Laravel 7.x.

symfony/console:

New Version: ^5.0

Reason for Addition: Laravel 7.x requires symfony/console as part of the upgrade for handling command-line operations and improving the functionality of Artisan commands. This package provides essential components for building command-line tools, which are crucial for tasks like running migrations, managing caches, and executing custom commands in the Laravel application. Symfony Console 5.x is needed to ensure compatibility with Laravel 7.x and to enhance Artisan command functionality.

illuminate/auth:

New Version: ^7.30

Reason for Addition: In Laravel 7.x, many components, including authentication and authorization, were separated into individual packages. The illuminate/auth package is necessary for handling authentication features in Laravel 7.x, such as managing user authentication, login, registration, and authorization. Since these components are no longer included by default in Laravel 7.x, adding this package ensures that our application can still use all necessary authentication and authorization features.

3. Removed Packages:

facade/ignition:

Removed Versions: ^1.11

Reason for Removal: In Laravel 7.x, the error handling package facade/ignition is no longer required as it has been replaced with nunomaduro/collision. The Collision package now provides more efficient and comprehensive error handling. It integrates better with Laravel 7.x, offering a better development experience. Since Collision is now the default error handler, facade/ignition is redundant and can be safely removed.

phpro/grumphp:

Removed Versions: ^1.0

Reason for Removal: phpro/grumphp is no longer compatible with Laravel 7.x and has been replaced by other packages that offer similar functionality for code quality checks and task automation. Removing this package helps maintain compatibility with Laravel 7.x, and using other alternatives ensures smoother integration with the updated version of Laravel.

roave/security-advisories:

Removed Versions: dev-master

Reason for Removal: roave/security-advisories was used to block insecure packages during development, but in Laravel 7.x, the framework includes built-in tools for

managing dependencies and security advisories. Laravel’s native tools make this package redundant, and it is no longer necessary to maintain the same functionality in the updated Laravel version.

Change from Exception to Throwable in Handler.php:

Reason for Change:
While upgrading to Laravel 7.x, we encountered an error due to a method signature conflict. The report() method in the Handler class was expected to accept Throwable instead of Exception. This caused an incompatibility with the base Laravel implementation. By replacing Exception with Throwable, we resolved this error and aligned the code with Laravel's updated error handling system.

Conclusion:

This upgrade process ensures that your project is fully compatible with **Laravel 7.30**, while also updating necessary packages and removing deprecated or unnecessary ones. By following these steps, you can leverage the new features, improvements, and better performance provided by **Laravel 7.30**, ensuring enhanced **security**, **maintainability**, and a smoother development experience.