Updating Exponential 6.0

Source file: `doc/eng-us/UPDATE.md` Last changed 2026-11-07
https://doc.exponential.earth/learninminutes/exphelp/viewer/UPDATE.md

This page updates a running Exponential 6.0 installation to a newer 6.0.x release or to the current state of the 6.0 line: back up, put the new code in place, update the database, rebuild the class map and the caches, check the result, and roll back if something is wrong. It is the short, practical procedure for developers and administrators. An installation older than 6.0 (4.x, 5.x, 5.90 or LoveStack) follows Upgrading, which has a part for each of them and ends with the same steps as this page.

> **Warning**
>
> `.htaccess` **slows down every request.** When `.htaccess` files are enabled (`AllowOverride` anything but `None`), let alone used, Apache looks for them and reads and evaluates them on every request, in every directory of the path, and page loading times go up drastically (measured). The only real fix is to switch the feature off for the virtual host: `AllowOverride None` in the HTTPS virtual host, and the rules in the virtual host's own configuration as mod_rewrite rules. That is far faster in production speed tests and well worth the time it takes to optimize a production configuration. Switch over right away: copy the rules of the example HTTPS virtual host into your virtual host (with the static cache, also `staticcache_offload.conf`), set `AllowOverride None`, run `apachectl configtest`, reload Apache and test the site. The same rules for other servers: nginx, LiteSpeed, Caddy, IIS, Velocity (built in); where virtual host rules go in a control panel: Plesk, cPanel, DirectAdmin, ISPConfig. **On LiteSpeed or OpenLiteSpeed,** the server's own page cache can also answer anonymous visitors and crawlers without PHP: Page caching with LiteSpeed and OpenLiteSpeed.

Every command is run from the installation root (the directory that holds `index.php`, `autoload.php`, `bin/` and `settings/`), as the user the site runs as. When you run a script as `root`, add `--allow-root-user`. `./console` is a shortcut for `php bin/php/console` (see Installing Exponential 6.0).

Contents

1. Before you start
2. Back up
3. Check the installation with the upgrade check
4. Put the new code in place
5. Update the database
6. Regenerate autoloads and clear the caches
7. Verify the update
8. Roll back
9. Related pages

1. Before you start

- **Find your version. Setup** > **System information** in the admin names it, and so does `php bin/php/console --version` (for example `console (Exponential) 6.0.15stable`). Keep the full system card for a comparison afterwards:

  Shell

  ```
  ./console exp:info --json --no-logo > before.json
  ```

- **Read the release notes of every version you pass.** They are in `doc/changelogs/6.0/` (one file per release, for example Changelog 6.0.15). Each names the database update, the scripts to run once and the changes of behavior to check. The checklist per period is in Upgrading, part C3.
- **Try it on a copy first.** Restore the backup of step 2 on a staging machine, update there, and only then update the live site.
- **Plan a window.** Maintenance mode answers every visitor with a 503 page while you work:

  Shell

  ```
  php bin/php/maintenance.php on --message="Back soon" --until=30m --allow-root-user
  php bin/php/maintenance.php status --allow-root-user
  ```

  See Maintenance mode.

2. Back up

The database and the files, taken together, with the site in maintenance mode. Use your own user, database and file names; `-p` makes the client ask for the password.

Shell

```
mysqldump --single-transaction -u USER -p DATABASE > backup-before-update.sql      # MySQL or Mar
iaDB
pg_dump -U USER DATABASE > backup-before-update.sql                                # PostgreSQL
sqlite3 var/storage/sqlite3/exponential.db ".backup 'backup-before-update.db'"     # SQLite: the
path of your database file
tar czf backup-files-before-update.tar.gz var settings/override settings/siteaccess extension des
ign composer.json composer.lock
```

Keep a record of the code you run now, so it can be put back exactly:

Shell

```
git rev-parse HEAD > backup-code-revision.txt      # a Git installation
cp composer.lock backup-composer.lock              # a Composer project
```

A backup you have never restored is only a hope: restore it once on another machine. More detail: Operating a site, section "Back up and restore".

3. Check the installation with the upgrade check

Before the update, find out what differs from the release you run, so that you know what the update will touch and which local changes you have to carry over.

- In the admin: **Setup** > **Upgrade check** (`/setup/systemupgrade`). **Check file consistency** compares every file listed in `share/filelist.md5` with its checksum; **Check database**

consistency compares the tables, fields and indexes with the schema files. Both checks only read.

- The file check from a shell, with the same findings:

  Shell

  ```
  php bin/php/checkmanifest.php --all --extensions
  ```

A file listed as **Modified** that you changed on purpose belongs in an extension or in `settings/override/`, not in the kernel; move the change there before you update, or it is overwritten. Each finding is explained in The upgrade check.

4. Put the new code in place

Choose the way you installed. Never let a tool overwrite `settings/override/`, `settings/siteaccess/`, your own extensions or your own designs.

A Composer project (the kernel is the package `se7enxweb/exponential`): set the version constraint in your project's `composer.json` if needed, then update the kernel package:

Shell

```
composer update se7enxweb/exponential
```

Composer-installed extensions are updated the same way, one package at a time, after reading their release notes (`composer update se7enxweb/<package>`; see `doc/changelogs/extensions/`).

> **Since 6.0.15** the kernel requires its libraries as **expcomponents** (`se7enxweb/expcomponents-*`), Exponential's fork of the Zeta Components, instead of `zetacomponents/*`. On the update to 6.0.15, run `composer update se7enxweb/exponential --with-all-dependencies` so Composer can swap the libraries; it removes each `zetacomponents/*` package and installs its fork. The classes are the same; nothing else changes (Upgrade notes: the switch to expcomponents).

A Git clone (the kernel is `https://github.com/se7enxweb/exponential.git`): fetch, then switch to the release tag or merge the branch into your own branch, then install the libraries `composer.lock` names:

Shell

```
git fetch --tags
git tag -l 'v6.0.*' --sort=version:refname | tail -5   # the newest releases; never sort tags by
name alone
git merge v6.0.14                                      # or: git merge origin/main, for the curren
t state of the line
composer install
```

Exponential Velocity is a Composer package of its own (`se7enxweb/exponential-velocity`) and is updated with the rest. After the code has changed, `composer.json` and `composer.lock` must match: `composer install` that ends without an error proves it.

Compare your files with the new templates. A release can change `.htaccess_root`, `.htaccess_root_static` and `config.php-RECOMMENDED`; your `.htaccess` and `config.php` stay as they are. Check them (nothing is written without `--apply`):

Shell

```
./console exp:setup:status                    # is .htaccess still a copy of a template?
./console exp:setup:htaccess                  # the diff between your .htaccess and the new temp
late
./console exp:setup:htaccess --force --apply  # take the new rules (backup first), when the diff
 holds nothing of yours
./console exp:config:list                     # settings new in config.php-RECOMMENDED show up a
s (not set)
```

Set the new settings you want with `./console exp:config:set <NAME> <value> --apply`; the settings you had keep their values. Use `--static` with `exp:setup:htaccess` when the site serves the static cache. See Setting up .htaccess and config.php from the console.

5. Update the database

The database update files are in `update/database/<engine>/6.0/`:

Engine	File for any 6.0.x to the current line
MySQL or MariaDB	`update/database/mysql/6.0/dbupdate-6.0.0-6.0.15.sql`
PostgreSQL	`update/database/postgresql/6.0/dbupdate-6.0.0-6.0.15.sql`
SQLite	`update/database/sqlite/6.0/dbupdate-6.0.0-6.0.15.sql`

Apply the file once, whole, in order:

Shell

```
mysql -u USER -p DATABASE < update/database/mysql/6.0/dbupdate-6.0.0-6.0.15.sql
psql -U USER DATABASE < update/database/postgresql/6.0/dbupdate-6.0.0-6.0.15.sql
sqlite3 var/storage/sqlite3/exponential.db < update/database/sqlite/6.0/dbupdate-6.0.0-6.0.15.sql
```

If your database already has a column the file adds, that one statement fails and names it; skip only that statement. A file that fails half way is never repaired by hand: restore the backup and apply the file again from the start.

Then run the one-time scripts the release notes name. For the 6.0.15 line they are in `update/common/scripts/6.0/`:

Shell

```
php update/common/scripts/6.0/createauditables.php
php update/common/scripts/6.0/movetrashrecords.php --remove-file
./console exp:user-login-check --allow-root-user                # lists logins shared by two acc
ounts
./console exp:user-login-check --fix-index --allow-root-user    # adds the unique index once the
re are none
```

`./console exp:checkdbfiles --allow-root-user` checks that the update files of your line are all present. The version row afterwards (the row names keep their historic spelling on purpose):

Shell

```
mysql -u USER -p DATABASE -e "SELECT name, value FROM ezsite_data WHERE name LIKE 'ezpublish-%';"
```

The complete description of this step, with the special cases of a site that came from 5.x, is Upgrading, part C2.

6. Regenerate autoloads and clear the caches

Rebuild the class map, then clear every cache, in this order:

Shell

```
php bin/php/ezpgenerateautoloads.php -e
php bin/php/ezcache.php --clear-all --allow-root-user
```

`-e` maps the classes of the extensions. When the kernel itself changed, add `-k`, and exclude any extra working copy of the code inside the root (for example a Git worktree folder) with `--exclude='<folder>'`, or kernel classes are mapped into that copy.

Then make the running PHP processes load the new code, and clear the caches in front of PHP last:

Shell

```
systemctl reload <your-php-fpm-service>                    # Apache or nginx with PHP-FPM
./console exp:velocity deploy --dry-run --allow-root-user    # Velocity: shows every step
./console exp:velocity deploy --allow-root-user              # Velocity: autoloads, caches, reloa
d, restart, in order
./console exp:velocity cache clear --apply --allow-root-user         # Velocity's response cache
```

Clear the content view cache **after** the reload (`php bin/php/ezcache.php --clear-tag=content --allow-root-user`): a page rendered by the old code between a clear and the reload would be cached again. After an update, rebuild the manuals of the help as well: `./console exp:help:content build-manuals --apply -s admin --release`.

7. Verify the update

Shell

```
./console exp:checkclasses --allow-root-user                 # every class loads under this PHP
./console exp:info --json --no-logo > after.json             # compare with before.json: every ve
rsion that changed
php bin/php/checkmanifest.php --all --extensions             # no file Modified or Missing that y
ou did not change
php bin/php/maintenance.php off --allow-root-user
```

Then in the browser: the front page loads, you can log in to the admin, **Setup** > **System information** shows the new version, **Setup** > **Upgrade check** passes both checks (your own deliberate changes aside), and a test item you publish appears on the site. Read `var/log/error.log` for anything new.

8. Roll back

Decide before you start at which point you roll back: when step 5 or 7 fails and the cause is not clear within the maintenance window. A rollback puts back the three things the update changed, in this order:

1. **The code.** Git: `git checkout` the revision in `backup-code-revision.txt` (or reset your branch to it), then `composer install`. Composer: put `backup-composer.lock` back as `composer.lock`, then `composer install`.
2. **The database.** Restore the dump of step 2 into an empty database (MySQL or MariaDB: `mysql -u USER -p DATABASE < backup-before-update.sql`; PostgreSQL: `psql -U USER`

`DATABASE < backup-before-update.sql`; SQLite: copy `backup-before-update.db` back to the database file with the site in maintenance mode). Never run the old code on the updated database: the two belong together.

3. **The files** in `var/` only if the update scripts changed them (`movetrashrecords.php --remove-file` removes `trashed.json`): unpack them from `backup-files-before-update.tar.gz`.

Then repeat step 6 (autoloads, caches, PHP-FPM reload or `exp:velocity deploy`) and step 7 with the old version. Write down what failed: the next attempt starts from it. A published release is never changed; a fix comes as the next version, so the way forward after a rollback is the next release, not a patched copy of this one.

Symptom	What to do
A white page after the code change	`var/log/error.log`; run step 6 again; `vendor/` must match `composer.lock` (`composer install`)
`Class ... not found`	`php bin/php/ezpgenerateautoloads.php -e`, then `php bin/php/ezcache.php --clear-all --allow-root-user`
A change does not show under Velocity	`./console exp:velocity restart --allow-root-user`, then `./console exp:velocity cache clear --apply --allow-root-user`
A SQL file failed half way	restore the backup of step 2 and apply the file again from the start
Libraries missing after a failed Composer run	Repairing an installation

9. Related pages

- Installing Exponential 6.0: a clean install of the same version, for a staging copy.
- Configuring Exponential 6.0: check the settings after an update, and which file wins.
- Extending Exponential 6.0: after an update, regenerate the autoloads of your extensions and run their upgrade steps.
- Upgrading: from 4.x, 5.x, 5.90 or LoveStack, and every checklist per period of 6.0.
- Uninstalling Exponential 6.0: the full data export, then the removal.
- The upgrade check and the file manifest.
- Setting up .htaccess and config.php from the console: compare both files with the new templates.
- Operating a site: backups, caches, cronjobs and repairs.
- Deploying and Velocity engines: reload the right PHP-FPM.
- The installation book: Upgrading.

Exponential Copyright © 1998-2026 Exponential Foundation and 7x
Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".
This documentation is free documentation, licensed under the GNU Free Documentation License 1.3 or later:
https://www.gnu.org/licenses/fdl-1.3.html.