



Please wait while we install a system update

Updating Neos – Why, When and How

Karsten Dambekalns

Karsten Dambekalns

- Developer and consultant at Flownative GmbH
- Working on & with Neos since 2005... for 19 years...
- Updated quite a number of Flow and Neos projects over the years
- Most recently a Flow 3.0.2 project essentially unchanged since 2013 🤪

Why, When and How

Why?

Because you get new features and bug fixes.

When?

Whenever a new version is released.

How?

Read & follow instructions, test & deploy.

Done!

Thanks for listening

- Any questions?
- Just kidding...

Why should you update?

- You benefit from fewer bugs, new features and better performance most of the time

Features to gain...

Greatly speed up node move actions

Due to an issue in
entities to comm
n^2 change comp

PSR 18 Http\Client

PSR 18 specifies a h

Atomic Fusion

Two years ago, W
Fusion" in a [blog](#)
It was trendy from
package has alrea

New Fusion Parser

The Fusion Parser has been rewritten from scratch
architecture with separate lexer, parser and an
continuously generated - not only line by line.

Fusion

This change
used just t

The idea is

Node Presets

Many developers are familiar
specific
This f
config

Lazy loading images

Image tags rendered with the *Neos.Neos:ImageTag* fusion prototype (or
the *ImageViewHelper* from the *Neos.Media* package) will now be rendered with the loading
attribute set
images until

Support of latest PHP versions

With the release of Flow 8 we are raising the minimal PHP Version to 8.0. This allows us to
take advantage of new language features especially the improved type system. With Flow
you can use typed properties, union types and php annotations. That way the code gets m
explicit, safe and needs less commenting.

```
1 rowClass = Neos.Neos:ImageTag
2 @subject = Read more a
3 one = 'd-
4 two = 'd-flex col-12 co
5 three = 'd-flex col-12
```


Bugs to get rid of...

4.3 and up:

- [BUGFIX: Fix drag-n](#)
- [BUGFIX: When co](#)

5.3 and up

- [BUGFIX: Make Mod](#)
[@gjwnc](#) in [ht](#)
- [BUGFIX: Add p](#)
[/neos/neos-de](#)
- [BUGFIX: Upda](#)
[/neos/neos-de](#)
- [BUGFIX: Fusio](#)
in [https://gith](#)
- [TASK: Add mic](#)

7.0

5.3 and up

- [BUGFIX: Exclude "/sites"-node from dimension-migration](#)

5.3 and up contain:

- [BUGF](#)
- [TASK:](#)

5.3 and up

- [BUGFIX: Rende](#)
- [BUGFIX: Impro](#)
- [BUGFIX: Prever](#)
- [BUGFIX: Fix two](#)
- [BUGFIX: Only u](#)
- [TASK: Simplify](#)
- [BUGFIX: Flush](#)

5.3 and up contain

- [BUGFIX: Only emit thumbnailCreated when successful](#)

Neos 7.3 Bugfixes

- Breaking Change [BUGFIX: Set the default for new shortcuts](#)
[@daniellienert](#) in [#3272](#)
- Breaking Change [BUGFIX: Always show current node in](#)
- [BUGFIX: Exclude "/sites"-node from dimension-migration](#)
- [BUGFIX: Make NodeSearchService use a new variable to](#)

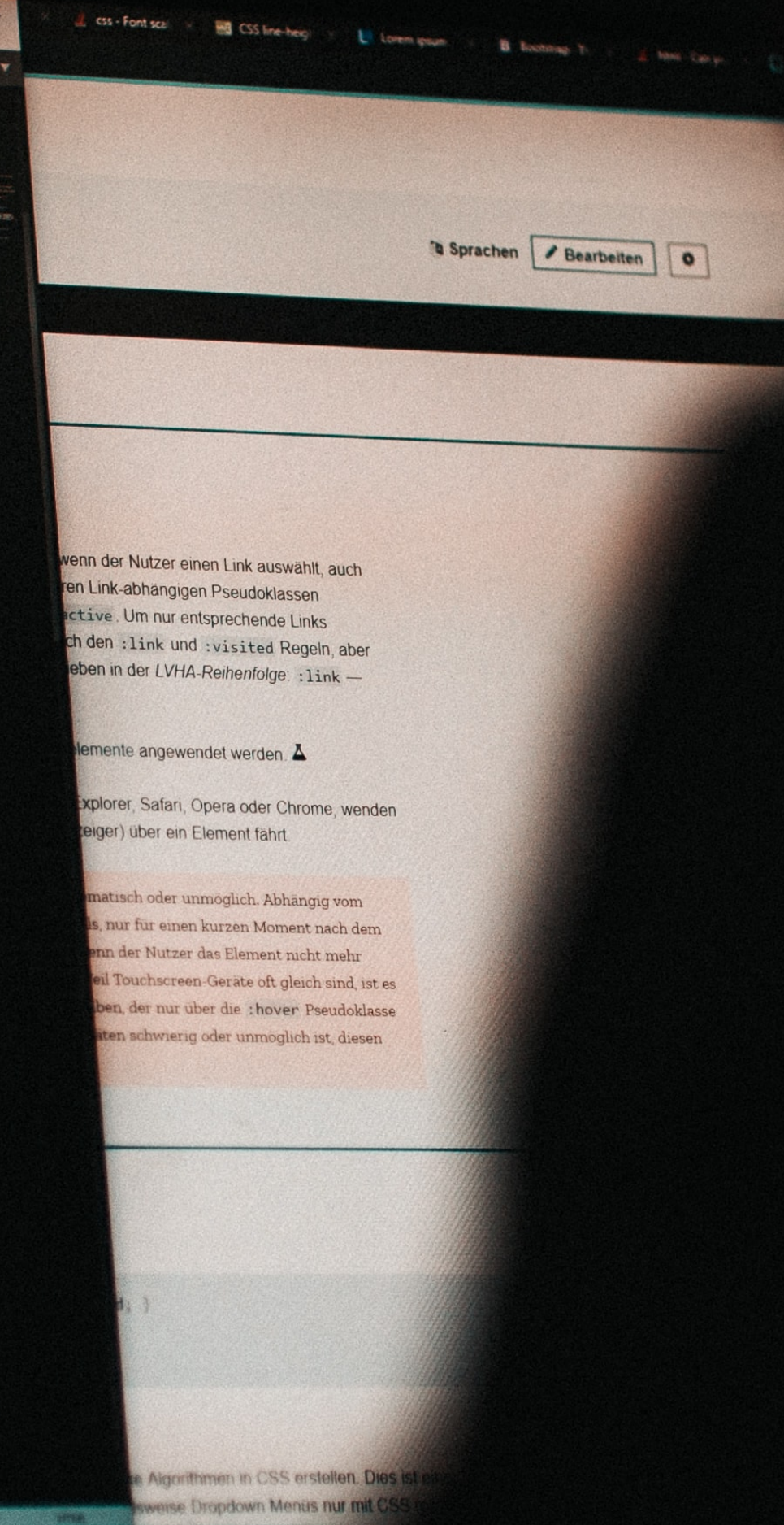
Why should you update?

- You benefit from fewer bugs, new features and better performance most of the time
- Outdated versions will no longer receive support

SECURITY

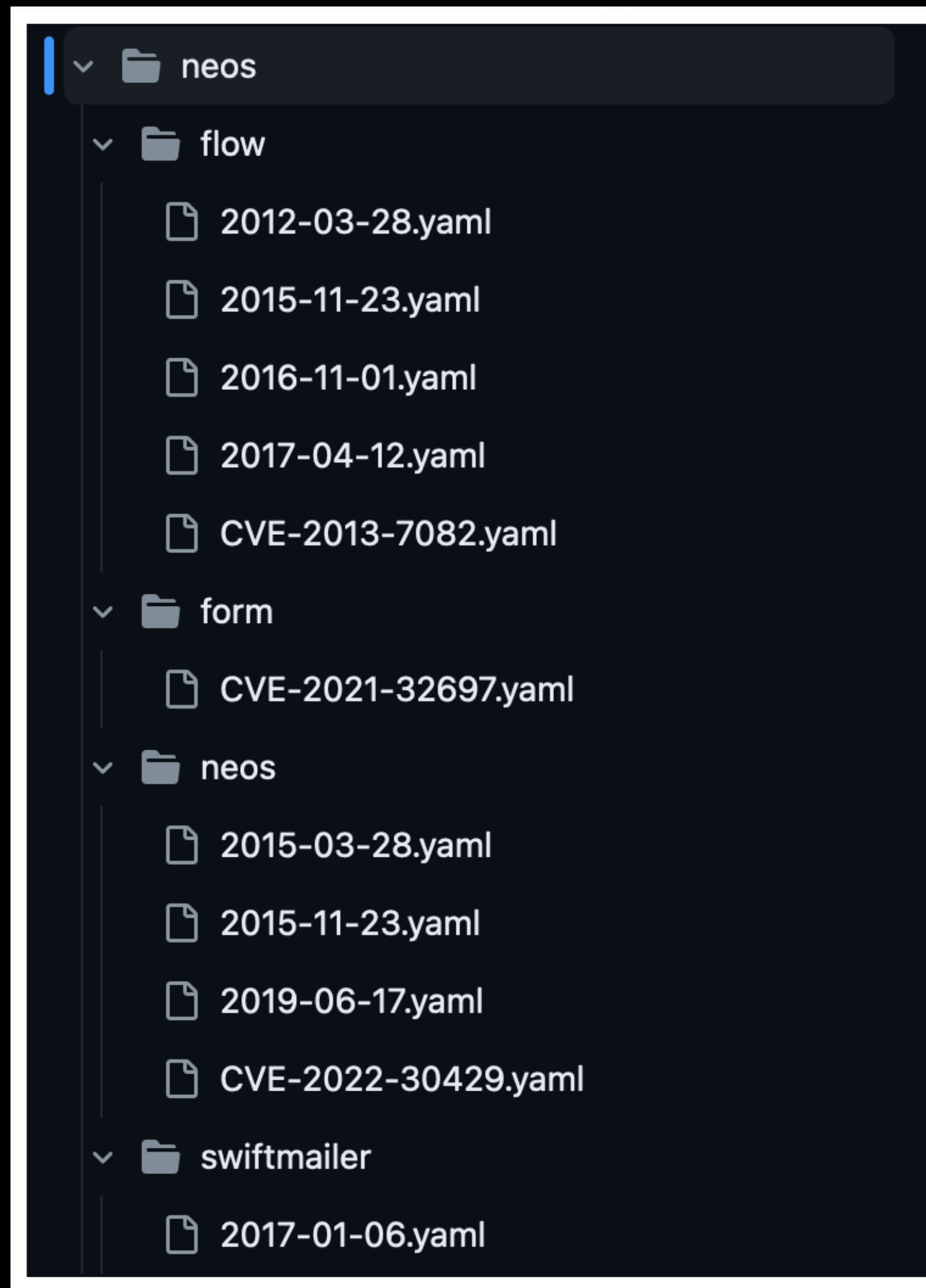



```
11 height: 100%;
12 }
13
14 .block {
15 width: 100%;
16 height: 500px;
17 margin: 0 auto !important;
18 padding: 0 auto;
19 line-height: 0;
20 }
21
22 h2 {
23 font-family: 'Montserrat', sans-serif;
24 font-weight: 900;
25 text-align: left;
26 font-size: 3000%;
27 z-index: 1;
28 transform: scale(-1, 1);
29 }
30
31 |
32 |
33
34 .column {
35
36 z-index: -1;
37 width: 20%;
38 display: block;
39 font-size: 400%;
40 }
41
42 </style>
43 <meta name="description" content="Tech-Texts by MB, the real
44 OS">
45
46 <meta name="keywords" content="Text">
```



Why should you update?

- You benefit from fewer bugs, new features and better performance most of the time
- Outdated versions will no longer receive support
- Security fixes will no longer be done on old releases



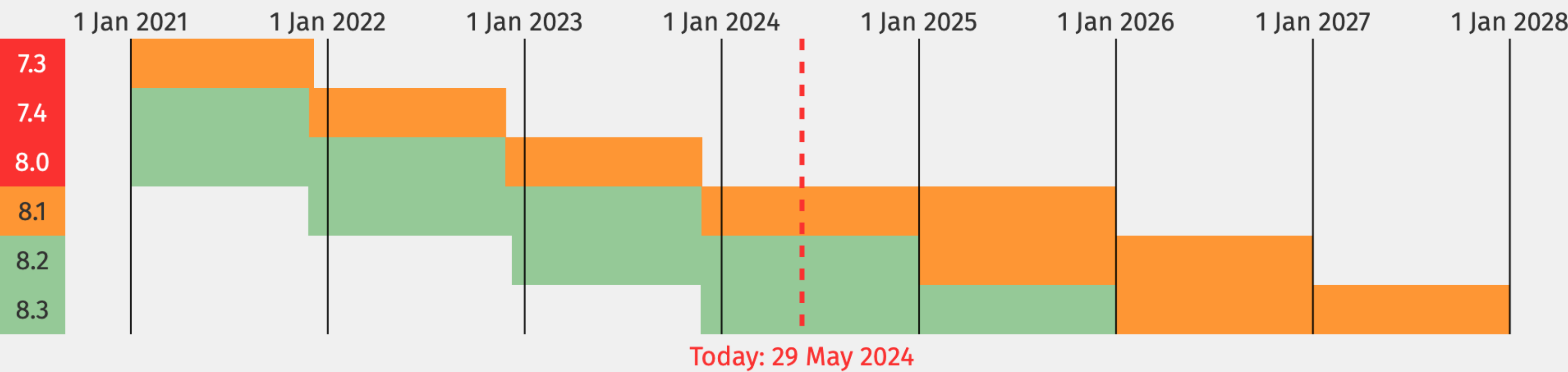
<https://github.com/FriendsOfPHP/security-advisories/tree/master/neos>

**Other tools
in your stack**

Currently Supported Versions

| Branch | Initial Release | | Active Support Until | | Security Support Until | |
|---------------------|-----------------|-----------------------|----------------------|---------------------|------------------------|----------------------|
| 8.1 | 25 Nov 2021 | 2 years, 6 months ago | 25 Nov 2023 | 6 months ago | 31 Dec 2025 | in 1 year, 7 months |
| 8.2 | 8 Dec 2022 | 1 year, 5 months ago | 31 Dec 2024 | in 7 months | 31 Dec 2026 | in 2 years, 7 months |
| 8.3 | 23 Nov 2023 | 6 months ago | 31 Dec 2025 | in 1 year, 7 months | 31 Dec 2027 | in 3 years, 7 months |

Or, visualised as a calendar:



Key

| | |
|---------------------|--|
| Active support | A release that is being actively supported. Reported bugs and security issues are fixed and regular point releases are made. |
| Security fixes only | A release that is supported for critical security issues only. Releases are only made on an as-needed basis. |
| End of life | A release that is no longer supported. Users of this release should upgrade as soon as possible, as they may be exposed to unpatched security vulnerabilities. |




Elasticsearch is a search engine that provides a distributed, multi-tenant-capable full-text search engine with an HTTP web interface and schema-free JSON documents.

| Release | Released | Support | Latest |
|---------|---|---|--|
| 8 | 2 years and 3 months ago (10 Feb 2022) | Yes | 8.13.4 (14 May 2024) |
| 7 | 5 years ago (10 Apr 2019) | Yes | 7.17.21 (03 May 2024) |
| 6 | 6 years ago (14 Nov 2017) | Ended 2 years and 3 months ago (10 Feb 2022) | 6.8.23 (13 Jan 2022) |

<https://endoflife.date/elasticsearch>

When should you update?

- As often and as fast as possible!
- Updating often keeps the routine drilled...
- Staying up to date keeps the steps to take smaller!
- Smaller steps mean lower risk when updating!



*An update a day, keeps
the downtime away*

But... we updated once, and it broke our site!

As mentioned, in case something goes wrong:

- do test your updates on a staging instance
- have a current backup
- have a rollback-strategy

If things still break on production deployment, the cause is almost never the new version...

But... we don't need new stuff and things work fine

That is the „our small site works well“ reasoning, which is fine until:

- Your hosting provider emails you to inform you about the EOL of PHP 5.6, so your Neos 1.0 site will stop working.
- Some security issue requires you to upgrade, but in a hurry!
- Related technology must be updated for some reason.
- New features **must** be added (GDPR adjustments, anyone?) – but no one wants to work with such old code...

But... we run a mission critical site!

Some sites cannot afford a downtime caused by a bug introduced by an update.

But of course you don't deploy the update to your mission-critical site untested, do you?

And you do have a backup and a rollback strategy in place, don't you?

If you can't have a staging instance for testing, your mission can't be that critical...

But... we only use LTS versions!

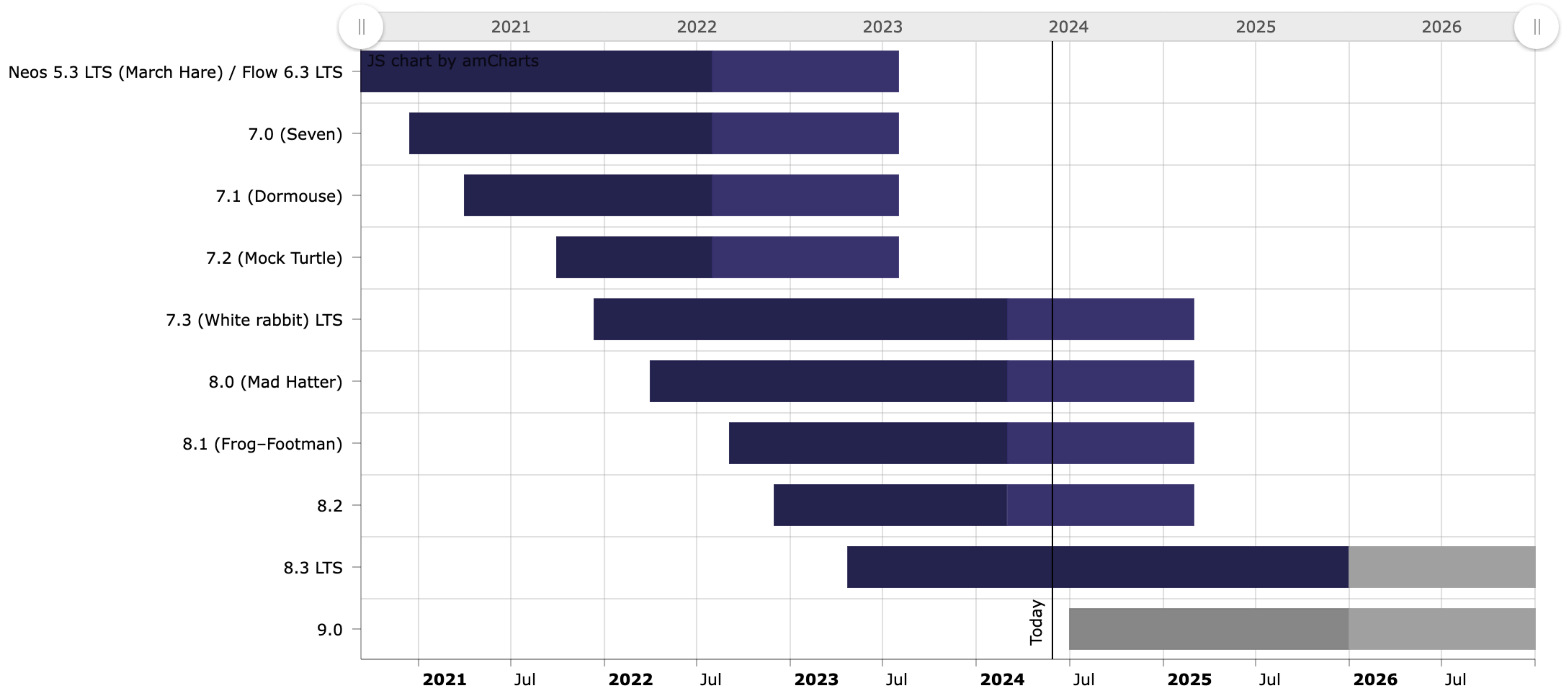
Sometimes people claim they must only use LTS versions, because they are more reliable or receive better support.

Reliability

Granted, new features may introduce bugs. But all supported branches receive all fixes – due to how we manage the source code.

Support lifetime

Any versions „after“ an LTS receive support for as long as the LTS version – due to how we manage the source code.



How should you update?

The generic recipe is:

- read release notes
- adjust composer.json, if needed
- do a dry-run of composer update, check results
- do composer update, commit result
- run core migrations, check result, adjust as needed
- run database migrations
- test, test again, commit, deploy to staging, test, deploy to production

Types of releases

Patch-level releases

- Must never be breaking, only contain bugfixes
- Might add new issues due to a buggy bugfix, happens rarely, though
- Usually about an hour of work

x.y.z+1

Minor releases

- Add features and contain all the features and fixes of the previous minor release(s)
- Must never be breaking, but new features may still have bugs (more probably in a x.y.0 release)
- Could take longer. If new features need to be enabled or configured, even a few hours, depending on the complexity of your site
- Might need a database migration and/or new configuration
- A bit more testing than for a patch-level release is advisable

x.y+1.z

Major releases

- Add features, contain features and fixes of the previous minor release(s) and may have breaking changes
- Probably need adjustments to your site's code and new (major) versions of used extensions
- Probably need a database migration and configuration changes
- Could require updated infrastructure (PHP or Elasticsearch version, ...)
- Needs the most testing of all types of releases

x+1.y.z

Approaching an update

Updating to a new version can always be done “in one go”, you do not need to upgrade to each version in between

But you always need to read the release notes of the full range of versions you cover in your update

Check your dependencies, if needed adjust to allow newer packages

Collect a list of (breaking) changes that need attention

Packages needing a dependency adjustment

Sometimes you must raise the allowed version of dependencies, e.g.

- flownative/google-cloudstorage 5.2 vs 5.3
- psmb/splitadd 0.3 vs 0.4

Usually not a problem, if you use constraints like ^5.2 – then 5.3 is allowed, too. And patch-level releases should always be allowed...

It's different for major versions (no surprise) and for 0.x versions!

Hint: use <https://semver.madewithlove.com/> to check constraints!

[Learn more about version constraints](#)

psmb/splitadd

^0.3

stable 

Results for **psmb/splitadd:^0.3**

0.4

0.3

0.2

0.1.1

0.1

dev-master

Satisfied?

```
composer require psmb/splitadd:"^0.3"
```

Doing the update

- As soon as your dry-run update works and gives reasonable results
- Update without the dry-run flag and commit the changes to the manifest(s) and lockfile
- Run core migrations on your own packages, check results, clean up as needed

Clean up after core:migrate

Here's what I do after running core migrations:

- check the commits that were created
- soft-reset the repository to the latest upstream commit
- adjust changes as needed
- commit the changes in one commit

That way it's easier for me to keep things clean and clear

Further adjustments

Core migrations usually adjust the „easy“ stuff, but some things cannot be automagically adjusted with reasonable effort

So some changes need to be done manually, examples for this are:

- **Fusion**
removal of the default prototype generator
- **Logging**
PSR-3 logger adjustments
- **Fluid ViewHelpers**
no more render() arguments
- **HTTP components**
use PSR middlewares instead

Neos 9

Upgrading to Neos 9

... is not just another major version upgrade.

Upgrading to Neos 9

Upgrading Composer packages

- Adjust the version of any Neos packages to require ^9.0 and run `composer update`.
- This will probably fail due to other packages in your project that need to be updated to work with Neos 9, so check those for newer versions.

You will also need to require (at least) one new package:

- `neos/contentrepositoryregistry-doctrinedbalclient` is the DBAL adapter for the CR

After the upgrade try to run `./flow` – with a bit of luck, everything compiles already. 🍀

Upgrading PHP code

- As usual, a first step is to run `./flow core:migrate` on all packages you maintain or want to create a PR for.
- Next install the `neos/rector` package to be able to use the migrations we ship for upgrading to v9.
- Now run Rector in dry-run mode and if things look good, run it for real.

If the automated ways of fixing your PHP code did not work (well enough), it is now time for manual adjustments...

- To begin with, just try to make the code compile again!

Migrating configuration

- If you have Routes configured, make them use the new FrontendNodeRoutePartHandlerInterface if needed.
- Adjust your Content Dimensions setup
- ... probably more to come!

Migrate your content

Moving your content from the old Content Repository into the new one is the major step.

In a nutshell:

1. Prepare your data
2. Prepare the new Content Repository
3. Install migration tooling
4. Do the data migration

Prepare your data

Your project ideally is at version 8.3 already (you upgrade early & often, no?)

To be on the safe side, run the migrations as usual:

```
./flow doctrine:migrate
```

To make the migration smoother, fix as many errors in the CR as possible.

So think about cleaning things up using `node:repair` to fix undefined properties, remove nodes with unknown nodetypes and so forth. This will make the output less verbose, later...

As usual, be careful with that tool, it can destroy data...

Prepare your data

Make your site's root node a dedicated "home page nodetype." Otherwise you will run into an error like this: The site node "bf...f8" (type: "Neos.NodeTypes:Page") must be of type "Neos.Neos:Site"

- If you already have a dedicated nodetype for the site root, you can simply add `Neos.Neos:Site` as a supertype to it.
- Otherwise, create a new nodetype like below and change the type of the current site root node to it:

```
'Acme.AcmeCom:Document.Homepage':  
  superTypes:  
    'Neos.Neos:Site': true  
    'Neos.NodeTypes:Page': true
```

Prepare the new CR

Before you can do anything, you need to set up the needed tables for the new Content Repository.

Fear not – this is as easy as:

```
./flow cr:setup
```

Now a number of new tables named `cr_default_*` will appear in your database – it is no problem to use the same database as before.

Install migration tooling

To migrate the data, you need another "helper package", so install `neos/contentrepository-legacy-nodemigration` now, which allows to migrate CR data:

```
composer require --dev neos/contentrepository-legacy-nodemigration
```

```
$ ./flow cr:migratelegacydata
```

```
Do you want to migrate resources from the current installation "/.../Data/Persistent/Resources" (y/n)?
```

```
Do you want to migrate nodes from the current database "dbname@dbhost" (y/n)?
```

```
Successfully connected to database "dbname"
```

```
Which site to migrate?
```

```
  [acmecom] acme.com (Acme.AcmeCom)
```

```
> acmecom
```

```
Site "acmecom" already exists, update it? [n] y
```

```
We will clear the events from "cr_default_events". ARE YOU SURE [n]? y
```

```
Truncated events
```

```
Exporting assets...
```

```
  Exported 211 assets. Errors: 54
```

```
Exporting node data...
```

```
  Exported 1840 events
```

```
Importing assets...
```

```
  Imported 196 Assets and 60 Image Variants. Errors: 0
```

```
Importing events...
```

```
  Imported 1840 events into stream "ContentStream:6ae1f48e-b3ed-4438-b685-d703c446cc29"
```

```
Replaying projections
```

```
Done
```

Migrate your Fusion code

- The automated migration using Rector will have adjusted your Fusion code already.
- Check those changes and look for instructions added – there are adjustments you **must** do manually yourself.
- Some more manual changes are needed when using deprecated ways of doing things, e.g. for adding @cache configuration.
- When (still) using Fluid, you need a few more changes – those doing AFX already are better off – as `Neos.Fusion:Template` is no longer a default.

Upgrading more PHP code

- Most projects will probably not be affected much by the changes to the Content Repository PHP API
- If you do interact with the CR, it is probably to
 - import things from some source,
 - export things to some source,
 - do things in some added Neos backend module.

As we come closer to the release of Neos 9, examples and recipes will be published for common cases!

If you get stuck

Finally, if you get stuck with an update even though you were well prepared

- Search for your issue, chances are someone else found a solution already
- Ask the community for help on [Slack](#) or discuss.neos.io
- Ask your agency for help
- If you are an agency, ask me or my company for help 😎

Test and deploy

Now test your site. You may

- Run unit and functional tests if you have them
- Run a link checker on the site
- Check logs for error messages and warnings
- Click around and manually test important or prominent features

If all is well, deploy and be happy. 🎉

Problems after an update

If some new problems appear after an update – ideally found before going into production – there are two possible causes:

- **You made a mistake.**
Double-check your changes and try to verify against the previous version.
- **You found a bug.**
Check the issue tracker for a new report that looks like your problem

Recap: Why, When and How

Why?

Because you get new features and bug fixes.

When?

Whenever a new version is released.

How?

Read & follow instructions, test & deploy.

Done!

Done. Questions!

Thanks for listening!

Further reading:

<https://www.flownative.com/en/blog/neoscon-2024-updating-neos.html>

Contact me at:

- karsten@flownative.com
- @kdambekalns in the Neos Slack



Flownative