Introduction for all stakeholders

Features and benefits of a 2.x upgrade

Modern and responsive user interface
Better versioning and asset handling
Flexible content and data architecture
Content blocks
Updated themes
Open Data integration through data.govt.nz
Pick-and-choose from recipe variations

CWP 2.x upgrade FAQs

What's the difference between an upgrade and a refresh?
What could go wrong during an upgrade?
Why should I upgrade?
What's the support cycle for 1.X?

SilverStripe CMS version support and CWP recipe compatibility
How long will an upgrade to CWP 2.x take? 8
How much will an upgrade to CWP 2.x cost? 9
How complex is my website when it comes to upgrading to CWP 2.x? 9
Is an upgrade an opportunity to improve? 9
What modules am I using? 10
Who can do my upgrade for me? 10
What will happen to CWP until 2020? 10
How are DIA and SilverStripe preparing for the future now? 10
What's the difference between the current CWP state and possible post-CWP future state? 11
How can my agency prepare for post-2020? 13

Planning a CWP 2.x upgrade for decision makers 14

Overview 14
Discover the benefits 14
Evaluate the risks 15
Assess your use case 15
Assess your quality assurance practices 15
Determine hosting options 15
Option A (recommended for high complexity sites) 15
Option B (recommended for low complexity sites) 16
Option C (alternative for low complexity sites) 16
Start a Project Plan 16
Get a quote and set a budget 16
Perform the upgrade 17
Project Plan for Stack Managers and Tech Leads

Overview

Estimating effort & risk
- Understanding the scope
- Resourcing
- Hosting and environments
- Modules
- Content blocks
- Files and versioning
- Custom features

Testing and training
- Manual testing
- Automated testing
- Visual regression testing
- Content author training

Development and migration
- Review the upgrade guide
- Upgrade modules
- PHP 7 Upgrade
- Follow the upgrade guide
- Prepare migration tasks
- Fulltext search reindexing

Go-live
- Preparation
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Testing</td>
<td>24</td>
</tr>
<tr>
<td>Switching over</td>
<td>24</td>
</tr>
<tr>
<td>Technical Guidance for Developers</td>
<td>25</td>
</tr>
<tr>
<td><strong>Overview</strong></td>
<td>25</td>
</tr>
<tr>
<td>Upgrade guide and changelogs</td>
<td>25</td>
</tr>
<tr>
<td>Upgrader automation tool</td>
<td>25</td>
</tr>
<tr>
<td>Considerations</td>
<td>26</td>
</tr>
</tbody>
</table>
Introduction for all stakeholders

The highly anticipated first major update of the Common Web Platform became available in 2018 with the 2.0 stable release. For content editors and technical teams in the public sector, this marks an exciting leap forward. 2.x minor releases will continue throughout the lifetime of CWP and you can jump right in to a 2.x upgrade.

The feature rich CWP 2.x takes advantage of the latest (and greatest) features from SilverStripe 4 and updates the CMS with major redesigns for content editors. For technical teams, it brings a refined approach to recipes and simplifies the maintenance of code.

An upgrade to CWP 2.x unlocks:

- A modern and responsive user interface
- Better versioning and asset handling
- Flexible content and data architecture
- Content blocks
- Updated themes
- CKAN Registry module
- Recipe variations

Below, we’ll dive deeper into some of the features and benefits included with a CWP 2.x upgrade and provide answers to some frequently asked questions. And later in this document, you’ll find planning advice and a draft Project Plan.

Features and benefits of a 2.x upgrade

Modern and responsive user interface

CWP 2.x is still the intuitive CMS loved by SilverStripe users, but simpler to use and visually cleaner. The main sections of the CMS are now responsive and reviewing/publishing content on different screen sizes have been simplified for authors.

We’ve also upgraded the WYSIWYG editor to TinyMCE 4 for better accessibility, powerful table editing, and a theme that’s much lighter on the eye.
Better versioning and asset handling

We’ve rebuilt the file management area from the ground up. Authors on CWP 2.x will enjoy a more intuitive interface and the ability to publish files alongside pages. What’s more, all content can have a draft stage, not just pages.

Flexible content and data architecture

The traditional approach to serving content and data lacks the flexibility to meet all audiences right at their doorstep. So with CWP 2.x, we’ve moved towards a more decoupled approach to publishing content allowing headless CMS applications with the introduction of new Content APIs.

This enables content teams, designers, and developers to deliver enhanced personalisation and greater flexibility in presentation across multiple devices and channels.

Read more: Decoupled body parts: headless CMS and SilverStripe

Content blocks

CWP 2.0 introduced the concept of content blocks into the CMS, enabling a new level of design sophistication and customisability for content teams. If you’re not already using blocks, this is an opportunity to improve the consistency of your site’s design, make it more modular and easier to maintain.

Read more: Let’s talk: Elemental

Updated themes

CWP 2.x added Wātea Plus to give agencies more control. Wātea is a versatile and visually appealing theme that is popular with Government sites. Based on feedback from the CWP community, new functionality now allows agencies to better customise the look of their site with minimal input from developers for increased alignment with their brand.

Read more: Introducing Wātea Plus

To support government agencies getting their sites operational quickly, in addition to the popular Wātea theme, CWP 2.x now includes a simpler Starter theme. This is a developer focused theme, catering to sites which require even more customisation and design.
Open Data integration through data.govt.nz

The CKAN Registry module introduces a new way of managing open data on your CWP site by displaying the information using a CKAN integration. For example, your agency might manage a list of contacts (such as the Family Services Directory) on data.govt.nz, and make this data searchable by visitors on your CWP website. With the module installed, Content Authors can easily create 'CKAN Registry page types' that are populated with your open data that is stored centrally on Data.govt.nz or another CKAN service.

Read more: CWP 2.2.2 featuring the new CKAN Registry module

Pick-and-choose from recipe variations

Your developers can now choose which combination of modules they pull in to each project. This modular approach saves your developers time, keeps unnecessary complexity out of your project, and therefore delivers projects faster with less cost.

CWP 2.x upgrade FAQs

Q: What’s the difference between an upgrade and a refresh?

A: An upgrade involves moving from one major version of a CMS to another (alongside upgrading any modules and custom code you may have). A refresh is a project that includes rethinking your website and how it achieves its goals. You can do both of these things at the same time, or just simply upgrade to CWP 2.x and call it a day.

Q: What could go wrong during an upgrade?

A: Since the modules which drive your custom code are changing, you need to understand and test functionality that’s specific to your website. Regressions can occur when there’s incomplete organisational knowledge of what custom functionality exists, or how it should work. Talk to your development partner or team about automated test coverage in order to drive quality assurance in your project.

Q: Why should I upgrade?

A: It’s the responsible thing to do for security reasons plus you’ll get the latest features. Being on the latest version means your site will continue to be supported. Participating agencies have
agreed to stay on supported versions, largely for security reasons. Upgrading to CWP 2.x is like upgrading your computer’s operating system—it’s a short term inconvenience for a longer term gain.

Q: What’s the support cycle for 1.X?

A: The CWP 1.9 recipe version will be supported until the end of September 2020. Refer to the CWP Releases and Changelogs for support timelines on specific release versions.

SilverStripe CMS version support and CWP recipe compatibility

Q: How long will an upgrade to CWP 2.x take?

A: It depends on the complexity of your website, upgrades range from several developer-days to developer-months.
Q: How much will an upgrade to CWP 2.x cost?

A: Contact vendors on the Web Services Panel (WSP) or your vendor of choice for an estimate that is sufficiently accurate for your budgeting purposes.

Q: How complex is my website when it comes to upgrading to CWP 2.x?

<table>
<thead>
<tr>
<th>Degree of code customisation</th>
<th>Low complexity</th>
<th>Medium complexity</th>
<th>High complexity</th>
<th>Very high complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Little to no custom code</td>
<td>Low to moderate amount of custom code</td>
<td>Moderate to high amount of custom code</td>
<td>Highly customised code base and complex integrations</td>
<td></td>
</tr>
<tr>
<td>Using Wātea theme</td>
<td>Some well maintained open source modules</td>
<td>Modules that haven’t been open sourced</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Using a standard ‘recipe’</td>
<td>Using a modified ‘recipe’</td>
<td>Using a unique ‘recipe’</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Q: Is an upgrade an opportunity to improve?

A: As your website evolves, existing code can become obsolete, non-performant, and unsecure. This means that the structure of your website can lose shape and become disordered.

Upgrading to a major version of SilverStripe is an opportunity for the structure of your website to address technical debt, enhance the experience provided to your customers, and update security measures.

A website before a major upgrade
A website after a major upgrade

When you upgrade to a major version it's an opportunity to...

Q: What modules am I using?

A: If you are on CWP 1.9, use Site Summariser to assess which modules you have. If you are not planning on upgrading to CWP 1.9, then we recommend you get in touch with your developers and request an assessment of your modules. Alternatively, SilverStripe can perform this activity for you through a Service Desk ticket.

Q: Who can do my upgrade for me?

A: Any development team familiar with SilverStripe development should be able to perform an upgrade. Web Services Panel (WSP) vendors are experienced with SilverStripe upgrades and they have capacity to do CWP 2.x upgrades. SilverStripe has additional partners outside of the Web Services Panel who can develop in CWP as well.

Q: What will happen to CWP until 2020?

A: SilverStripe and DIA will keep improving CWP until the existing agreement comes to a close at 16 September 2020. Over 2019 and 2020, SilverStripe will develop the platform aligned to a shared roadmap that is co-developed with DIA and participating agencies.

Q: How are DIA and SilverStripe preparing for the future now?

A: DIA intiated a Web Content Management/Digital Experience Platform Project where they worked with agencies and suppliers to co-design the next iteration of CWP. Your agency will
likely have been involved in a series of workshops to provide input. For more information about this, we recommend you contact Kayleigh Shepherd: Kayleigh.Shepherd@dia.govt.nz

SilverStripe has a clear vision for the future of our products and how they can help the New Zealand government to transform the way it designs and provides digital experiences and services to Kiwis. We will use our expertise in supporting stable, high performing public sector websites—and the ecosystems surrounding them—to keep helping agencies further place citizens at the centre of government digital services delivery.

Q: What’s the difference between the current CWP state and possible post-CWP future state?

A: The below two diagrams show the CWP current state and possible post-CWP future state. Underneath CWP—and fundamental to the CWP product—is SilverStripe. We are here to stay, continuing to be your flexible, world-class partner in your digital platform journey.

**Service Composition - Current state**
**Possible Future state (post-2020)**

SilverStripe are committed to supporting DIA in the delivery of post-CWP solutions. We are actively involved in the co-design process with DIA and once DIA’s plans are more fully formed, we will finalise our approach to supporting CWP clients post-CWP.

In the interim, have confidence that your investment in CWP, and service continuity beyond 2020, is assured by SilverStripe’s commitment to maintain an offering that is functionally equivalent.

This means that whether or not there is a formal role for SilverStripe in providing future services via DIA, we can offer to take on additional responsibilities to ensure that CWP clients retain their existing functionality, performance, security, and price, with a minimal transition cost.

A possible future state involving SilverStripe might look like this:

---

An illustration only. Depending on a number of factors, this may not be representative of SilverStripe’s post-CWP offering. Standard rules of procurement to be followed.
Should DIA seek another way of working with CMS vendors, such as the Marketplace, we would support this and offer our replacement service through this channel.

Q: How can my agency prepare for post-2020?

SilverStripe are committed to providing continuity of innovation and support for CWP sites beyond DIA and SilverStripe’s agreement ending in September 2020. We will support agencies who continue with our tailored Platform option catering to the needs of the New Zealand Public Sector. Rest assured that your investment in CWP is secure well into the future.

We recommend that you upgrade to CWP 2.x as soon as is practical. SilverStripe will be able to transition CWP 2.x clients to a future state with minimal effort, providing continuous support post-September 2020.

We are fully committed to delivering the leading CMS and full-service hosting platforms designed to serve the evolving needs of your agency. Until September 2020, this will be accomplished within CWP. From October 2020 onwards, we will ensure that we can continue accomplishing this, either through collaboration with DIA, or through the SilverStripe managed platform.
Planning a CWP 2.x upgrade for decision makers

Overview
Here’s a quick start checklist to get the ball rolling. There’s a more in-depth guide to create a “Project Plan for Stack Managers and Tech Leads” below.

Discover the benefits
Consider the specific benefits of upgrading depending on the version that your site is currently on. You can find out what version you are on by asking your developers to use the CWP site summariser or checking your environments on the deployment dashboard.

Then ask your developers to look at the changelogs for the newer versions to see what features you’re missing out on!

For example, if you find you are on CWP 1.9 / SilverStripe 3.7, look at the changelog for the next stable release (e.g. CWP recipe basic 2.0.0). And if you are on CWP 1.6 / SilverStripe 3.6, look at the changelogs for the 1.7, 1.8, 1.9, 2.0, 2.1, 2.2... stable releases. The further behind you are, the more you can gain by upgrading.

To see a full list of CWP changelogs and the corresponding SilverStripe framework version, see the CWP releases and changelogs page.
Evaluate the risks
What modules does your site depend on? Are they up to date? If you are using non-recipe modules, ensure the site has been penetration tested. Use the Site Summariser if you are on CWP 1.9 or above. Alternatively, talk to your developers or log a Service Desk ticket. Try to familiarise yourself (at a high level) with the process of upgrading a CWP/SilverStripe site. In order to deliver the benefits of CWP, upgrading your CWP site is not like upgrading a simple SaaS product. Make sure you have a technical leader to assist you and make sure they read the sections relevant to them.

Assess your use case
If you are on CWP 1.9, use Site Summariser to assess which modules you have. As covered in the FAQs, the modules you are using affect the complexity of your upgrade. At the low complexity end of the scale are sites using the Wātea theme modules. At the high end of the scale are sites using modules that haven’t been open sourced.

Assess your quality assurance practices
Do you have existing automation tests (unit, integration, end-to-end)? Are they currently passing, and do they provide sufficient cover for the features important to your agency? This might be a good opportunity to talk to your developers about future proofing your site by building up testing practices (see our guide).

Determine hosting options
Each CWP environment has a UAT and Production environment. During the upgrade, you'll likely want additional environments in order to prepare and test upgrades. In some cases, upgrades involve long-running migration tasks, especially if your website has a lot of assets. There can be database and search index changes which take some time to roll back in case of errors. Depending on your risk assessment, there are a few options for provisioning environments for your upgrade.

Option A (recommended for high complexity sites)
Set up a new parallel Stack that is the same as the existing site, load up a copy of the site and all the assets, and update there. Once you know you are good with everything, transfer over refreshed assets, then switch the domain to point to the new Stack and you are done!
Pros: No downtime as you are building in parallel, less impact (i.e. data and cost) if there are issues, no expensive out of hours go-lives, and less overall risk.
Cons: Paying for two Stacks at once.

Option B (recommended for low complexity sites)

Set up a new environment on the existing stack, deploy through to the existing UAT and Prod.

Pros: Less expensive at the beginning as a new environment costs less than a Stack, much faster/easier to transfer around code and assets.
Cons: Higher risk and potential for costs to be greater than setting up a new Stack; downtime during go-live; potential out-of-hours costs if needing to be done outside of business hours.

Option C (alternative for low complexity sites)

Use existing Test or UAT environments for the upgrade.

Pros: No cost impact, no reliance on CWP Service Desk.
Cons: Higher risk for go-live (see Option B). Hard to test changes to your existing CWP 1.x website without restoring asset and database state prior to the in-flight migration on that environment.

Any temporary environments required for CWP 2.x upgrades don’t incur setup fees in the CWP Service Desk and you can cancel them once they’re no longer required. New parallel Stacks do incur a setup fee to cover the additional configuration needed to make it production-like but the monthly fee is considerably discounted.

Start a Project Plan

Work with your vendor or in-house dev team to create a project plan (see our “Project Plan for Stack Managers and Tech Leads”).

Get a quote and set a budget

Ask your vendor or in-house dev team for an estimate and budget accordingly. Factor in the costs for development plus hosting. Schedule an upgrade with your in-house team or vendor of choice.
Perform the upgrade

Your in-house dev team or your vendor will run a practice deployment, test it, and then go through UAT testing and bug fixes. Tests should be at a code level and at a user level. The “Project Plan” has more guidelines on what to consider during testing.
Project Plan for Stack Managers and Tech Leads

Overview

This guide will help Stack Managers to collaborate with vendors and their development teams to create a project plan.

Estimating effort & risk

Understanding the scope

Your development team or vendor will need to be aware of what has changed in CWP 2.x, and how that affects your codebase. Particularly if this is their first upgrade, it’s important to read through our “Technical Guidance for Developers” during the estimation phase, which includes references to changelogs and technical migrations.

Resourcing

The roles required for an upgrade project aren’t different from any other CWP or website project. If you have a large custom codebase, it might be worth starting with an exploratory upgrade to de-risk the process. In the early stages, it can also be advisable to keep the number of developers low. Until they’re getting the project to “dev/build” on CWP 2.x, it’s quite likely that multiple developers will end up fixing the same bugs and getting into each other’s way. Once they’re over this initial hurdle, the upgrade work can be more easily distributed to multiple team members.

Hosting and environments

We recommend provisioning at least a separate CWP environment for upgrades, keeping your current UAT environment on CWP 1.x. This retains your ability to quickly deploy hotfixes on your existing site, without switching branches or restoring database and asset states. In many cases, it is advisable to build up a parallel CWP stack with a separate production environment which you can safely switch over to during go-live. Any temporary environments required for CWP 2.x upgrades don’t incur setup fees in the CWP Service Desk, and you can cancel them once they’re no longer required. New parallel Stacks do incur a setup fee to cover the additional configuration needed to make it production like but the monthly fee is considerably discounted.
Since upgrades involve file migrations, you should ensure that you have sufficient storage space for database and asset snapshots. You can find out how much snapshot space is available in your dashboard, and can determine how much space is required by creating a snapshot. In order to restore snapshots, you also need to ensure that enough disk space is available on the target environment to extract it, in addition to the existing assets. We recommend contacting the CWP Service Desk when planning the upgrade, and discussing adding temporary environments or disk space capacity to your stack.

Modules

You’ll need to know which modules your website uses, which are supported, which are custom, and which require work to be compatible with CWP 2.x (SilverStripe 4.x).

The majority of modules in the supported modules list have SilverStripe 4.x compatibility. This can usually be determined through the module README. Your developers will be able to track down the current module by searching on addons.silverstripe.org. Alternatively, you can run the first step of the upgrader tool and get a list of potential problems automatically. If the module doesn't appear to be upgraded yet, make sure to check any open issues or pull requests (see “Maintaining your Code” and “Using Module Forks” on cwp.govt.nz). You can also search for known issues, report module bugs, or contact the CWP Service Desk if you’re not sure.

If you have custom code that could be useful to other agencies, this is a good time to talk to your developers about potentially open sourcing it as a module (see “Code Sharing” on cwp.govt.nz).

There is a small number of modules which are only supported in CWP 1.x (SilverStripe 3.x), and not in CWP 2.x (SilverStripe 4.x). In some cases, that’s because the related functionality has been folded into other parts of the CWP recipe. Please review the “SilverStripe 3.x only support” section of the supported modules list and check with your developers if this has an impact on your project.

Content blocks

CWP 2.x introduces content blocks as a way to provide more flexibility for authors in creating and structuring content. There’s also a greater potential for reusing existing code from the SilverStripe community, and a potential to reduce the custom code you maintain. Content blocks are a great opportunity to establish a pattern library on your site for a consistent user experience and less design variations to maintain. This gets you closer to a redevelopment of
your site, but the payoffs for both authors and developers can be worth it. Read more about Content Blocks in the CWP 2.0 release announcement, and in our recent blog post “Let’s talk: Elemental”.

Files and versioning

Versioning of content has been implemented throughout CWP 2.x, most notably changing how the “Files” section works. If you’re currently making heavy use of file versioning, or file version history is critical to the website, then you will need to factor in some extra time for planning how you do the migration of files. Otherwise, there is a built-in file migration task which can transform from CWP 1.x to CWP 2.x compatibility (but please note that this will make existing version history of your files inaccessible through the CMS UI).

With any migration of files, you should also assess how much disk space is currently being used. A low volume of assets can be migrated locally then pushed through UAT and finally back to production during go-live, but for larger volumes there are a few things to consider:

1. The CWP snapshots feature is instrumental to migrate data and assets between environments. If you have access to a new parallel product environment on a separate stack, you can also move snapshots between stacks without downloading them. Long-running migration tasks are supported in CWP via the queuedjobs module which is built into the CWP recipe. See “Hosting and environments” for more infrastructure considerations.
2. If you download snapshots for a local migration, they need to be uploaded into CWP again. Snapshots over 256MB require Service Desk support. This may take a few days and would potentially require you to provide a physical copy of the files (i.e. USB).
3. If your stack has enabled the Disaster Recovery (DR) option, it may take some time for your assets to sync between data centre locations and you should factor this in to any go-live plan.
4. If your stack is using a large amount of disk space or has a complicated migration process, you may want to consider creating a parallel Stack running alongside your current Stack, with separate environments (including a new production environment). You can then do the migration in the background and switch over when you’re ready, with assistance from the CWP Service Desk. Some guidelines are available in our File Migration Guide.
Custom features

As CWP 2.x (and SilverStripe 4.x) is a major release with backwards incompatible changes, you will need to upgrade your own code to be compliant. The changes you need to make will depend on the size of the codebase, the non-standard features of the website, and custom integrations that your website may have. Your development team should be able to make an informed estimate based on the specifics of the website’s needs. Tip: You can get an overview of the size of your PHP codebase through **phploc**.

Testing and training

Manual testing

Regression testing is arguably the most important part of any upgrade. While developers are able to do some testing as they wade through lines of code, a professional tester is able to take a step back and work closely with stakeholders to formulate a test plan. You should consider seeking the services of professional testers.

Automated testing

If you have already invested in automated tests for your website then you’re about to reap the rewards as they act as a safety net, catching regressions during the upgrade that would have otherwise been missed.

If you haven’t yet invested in automated tests, then now is a good time as this will provide an automated way to test critical features before you upgrade. Testing before the upgrade means you have something that tells you with reasonable certainty that your website does what you expect it to. Running them after the upgrade will alert you if something has been broken meaning you can catch bugs before pushing to production.

When deciding what features should be covered with automatic tests, you should prioritise the most critical features first and any automated tests that you’re relying on should pass before the upgrade as well as after. Check out our "Unit Testing" guide, as well as NightwatchJS and silverstripe/testsession as a common combination for end-to-end testing.

Visual regression testing

A cost-effective way to spot visual regressions caused by the upgrade is to run a visual regression testing tool like **wraith** or **Ghost Inspector**. They can crawl your site before the
upgrade, taking screenshots of what each page looks like and then again after the upgrade. Finally, it will compare the screenshots and highlight any pages with differences.

Content author training

While the CMS will have a very similar look and feel, some aspects of it may change after the upgrade. At the very least, you should plan to give content authors time to familiarise themselves with the CMS.

Training is available to agencies and suppliers to quickly get people building and maintaining websites on the Common Web Platform (CWP). Training is delivered by SilverStripe Ltd (dates and costs).

This training includes Content Management System (CMS) training for editors and administrators using the New Zealand Government Common Web Platform. If you seek training for a website that has been modified from the standard CWP install, or seek training for a SilverStripe CMS website not running on the Platform, please get in touch with SilverStripe to discuss options. Content editor training is a 3 hour course, designed to give non-technical communications and business staff the general skills to effectively use the SilverStripe CMS for publishing agency websites. Courses focus on either content editing or administration tasks based on learning requirements.

Development and migration

Review the upgrade guide

Understanding and following the SilverStripe 4 upgrade guide is the most important thing to be aware of when beginning development. Using the upgrade guide and upgrade automation tool will make your job a lot easier as it takes care of some of the code upgrade for you automatically. See our “Technical Guidance” specific to CWP.

Upgrade modules

Any modules which aren’t CWP 2.x (SilverStripe 4.x) compatible will need to be upgraded (see our guide). It’s best to do that at the start of the upgrade, in isolation from the project if possible. This allows you to test the module in isolation and ensure any problems can be found
quickly and easily. Depending on how deeply integrated the module is, you could also choose to leave out the module initially and make some headway with the step-by-step guide below.

**PHP 7 Upgrade**

Depending on when you started your CWP 1.x site, it might have been built on the PHP 5.6 language, which is now end-of-life. You should upgrade your site to PHP 7.x, which is usually a small task. And it has an upside: both your CMS and site will likely respond faster. See our [PHP 7.1 upgrade guidance on cwp.govt.nz](https://cwp.govt.nz).

**Follow the upgrade guide**

After you’ve reviewed the upgrade guide and you have all modules upgraded, you’re ready to start stepping through the [upgrade guide](https://cwp.govt.nz). If your website relies on the CWP basic recipe, use the dedicated ‘--cwp-constraint’ parameter for the recompose command otherwise the recompose will try to upgrade your composer file to the latest SilverStripe 4.x compatible versions of modules.

**Prepare migration tasks**

As part of the upgrade, you’ll need to consider how you get your data from CWP 1.x to 2.x. The framework includes a file migration task which will help you migrate to the new file system format.

Depending on which modules you use, there may be other similar tasks providing pathways for an easy upgrade. However, you should check each module and inspect the migration task to decide whether this will work for your project since there may be project customisations which the task won’t expect.

**Fulltext search reindexing**

CWP bundles the Solr fulltext search module. If your site uses this to search website content, an additional step might be needed. The module relies heavily on class names which can change with the upgrade if you’re opting to namespace your custom code. Set aside time during the migration for Solr to reindex and ensure the site can function without Solr if necessary for a short period of time.
Go-live

Preparation
You should ensure that you have a clear plan for go-live, including a plan to roll back should something go wrong. Provisioning a separate CWP environment for upgrades retains your ability to quickly deploy hotfixes on your existing site, without switching branches or restoring database and asset states. We recommend minimising the non-essential development work on the existing site while the upgrade is in progress. Read through the “Managing Deployments” guide on cwp.govt.nz.

Running through the go-live on UAT or another parallel CWP stack will highlight risk areas and where support may be needed from CWP Service Desk. If you find that you need this support, you should raise a ticket with the CWP Service Desk who can advise you and your team.

Testing
Before going live, you should test the features which are custom to your website. Some examples:

1. Search returns expected results
2. Login works and you can use the CMS as expected
3. All published files and images appear on the site

Because SilverStripe sites can be customised quite heavily, you should work with your content authors, testers, and vendors to compile a testing plan which makes sense for your context.

Switching over
If you have chosen to build your upgraded site on a parallel CWP Stack, the go-live can be as simple as switching a DNS entry (see “DNS/Go-Live” on cwp.govt.nz). If you are upgrading a site in-place (on the existing Production environment), work with your vendor or in-house development team to schedule downtime during the upgrade. Every CWP site has a built-in maintenance screen, which you can use to redirect visitors during an upgrade. There are ways to bypass this screen for certain IP address ranges, which can be useful to test the new site under production before activating it for all visitors. Please talk to your developers about setting this up in an .htaccess file.
Technical Guidance for Developers

Overview

This guidance expands on the high-level “Project Plan for Stack Managers and Tech Leads”, and goes into more detail on the technical steps and considerations required to perform an upgrade of your site to CWP 2.x. It is aimed as a starting point for developers.

Upgrade guide and changelogs

CWP 2.x is built on top of SilverStripe 4.x, but has additional functionality and modules. This means you need to read up about changes for both CWP and SilverStripe releases. Start off by reading about the high level technical changes in the SilverStripe 4 upgrading guide. The main breaking changes are described in the CWP 2.0.0 changelog and SilverStripe 4.0.0 changelog. Since then, we have been busy releasing new minor versions of both CWP and SilverStripe 4.x. Some of these releases add new functionality. For example, SilverStripe 4.1.0 added support for a public/ folder, and CWP 2.1.0 added an installed modules report. You should always upgrade to the latest release, but those introduce non-breaking and opt-in changes only (our modules follow semantic versioning). You can see a full list of SilverStripe 4.x changelogs and CWP 2.x changelogs, and subscribe to the CWP Newsletter to stay in touch when new releases come out.

Upgrader automation tool

A lot of the upgrade effort comes down to syntax and naming changes, which we've mostly automated through an upgrader tool. It will assist you to:

- 'recompose' your requirements
- update your environment settings
- namespace your code
- update your code to use new APIs
- become aware of issues by alerting you to any issues it may identify
- relocate your web root assets
- re-organise your project layout (more relevant with upcoming releases)
## Considerations

There are a lot of decision points around upgrades which are less clear than renaming PHP classes. We’ve summarised these in the “Project Plan for Stack Managers and Tech Leads”. Here’s a quick summary about the topics you should discuss with your stakeholders leading up the technical upgrade process:

- Are there module upgrade paths for all modules we’re using?
- Are there modules on addons.silverstripe.org which can replace custom code?
- Can we identify unused functionality and remove it?
- PHP 5.x is end-of-life. Do we need to upgrade to PHP 7.x?
- Should we allow more flexible content templates by rewriting our templates to use content blocks?
- Should we start versioning more of our content, in order to give authors greater control over drafts and previewing?
- Should we declare ownership between versioned relationships, in order to “cascade” publication events?
- Uploaded files aren’t published by default. Do we need to create ownership on objects with file relationships?
- Should we refactor any custom many-many relationships to a more flexible many-many-through format?
- Can we use the more granular recipes to trim down or expand the modules we include in the project?
- Can we increase our quality and stability by adding automation tests (e.g. via NightwatchJS and silverstripe/testsession) or visual regression tests (e.g. via wraith or Ghost Inspector)