Our journey scaling WordPress for NewsCorp Australia

Coded for WordCamp Europe Vienna 2016

18 months ago, in Sydney, Newscorp Australia started it's journey to migrating all of websites over to WordPress.

Both Juan and myself have been part of the software engineering team working on this migration project.

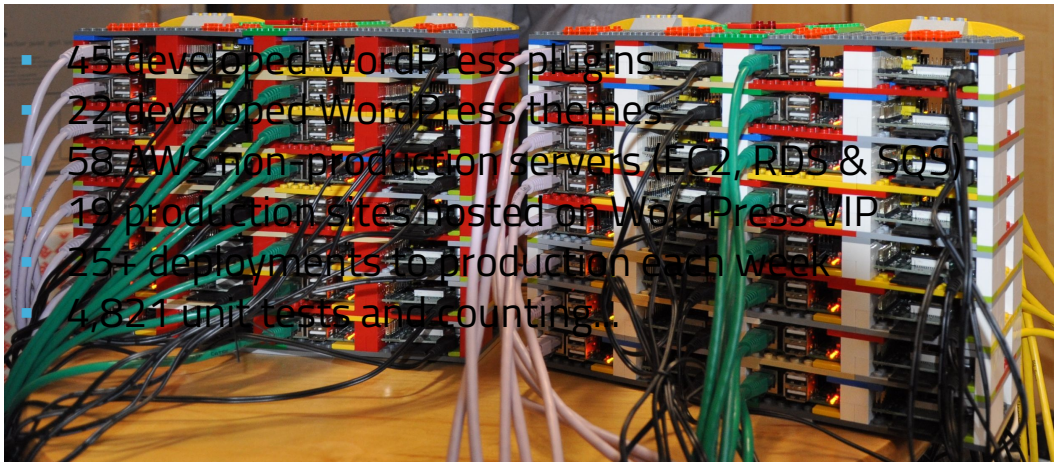
You often hear people say "WordPress doesn't scale", our talk today is to prove them wrong and to demonstrate that WordPress can scale for enterprise size businesses

- Historically people may have interpreted Newscorp looking a little bit like this, focusing on the traditional print newspapers.
- Yes our company has been around for over 90 years, we are Australia's largest News & Media company and part of NewsCorp global.
- Over the course of the last 18 months, we migrated 19 of Australia's largest traffic sites including news.com.au, theaustralian.com and foxsports.com.au to WordPress VIP
- We have imported around 4 million news stories into WordPress and have approximately 18 million visitors a day

Why did we move to WordPress VIP?

- If you are not sure about what WordPress VIP is, it is an enterprise hosting option for WordPress sites hosted on the wordpress.com infrastructure.
- So we needed to escape our old, unscalable, expensive CMS. WordPress was chosen for many reasons, but primarily because it provides a great authoring experience that most of our editors/journalists had already used (making training and upskilling easier). We just had to find a way to scale wordpress to meet the needs of our business.
- Lastly, possibly the best part, (click), this does not happen to us, WordPress VIP allows us to let wordpress.com manage our production infrastructure so we can focus on writing code and delivering great new features to our customers - we no longer have to worry about keeping our WordPress stack up to date etc…

What does our setup look like?



- So at a high level,
- We have developed 45 custom wordpress plugins along with 22 custom themes to power our sites.
- Apart from our production infrastructure, we run 58 AWS non-production environments (which are powered by a combination of EC2, RDS & SQS).
    - Which mimic the VIP setup as much as technically possible
- From a deployment perspective we do around 25+ deployments to production each week through our CI/CD platform.
- And we love our unit tests, nearly 5,000 to date

- Our journey started in November 2014, and when we looked back at the last 18 months we had been through with our migration to WordPress VIP, we identified 5 key factors that caused our team the most challenges. We wish we had of known about before we started the project. However, encountering these challenges and discovering their solutions was part of our journey (and a great team building experience). But… it was a tough tough journey.
- So we want to share our story for 2 main reasons:
    - One, to demonstrate that with work, wordpress can scale for the enterprise.
    - And to share the challenges we faced in scaling wordpress and our solutions so other companies can learn from our migration challenges.

1. Site build
2. Front end
3. Content
4. Authoring
5. CI/CD

- We are going to touch on these 5 key areas… Talking about the challenges that we faced in each area and the solutions that got us over the line..

Nov 2014 Feb 2015

1.
Site build

Enhancing customizer

- It all starts with site build, we needed our wordpress admins to have a simplistic workflow to organise the content on our websites.
- Customizer in WordPress was the start of the answer to this.
- It provided a great base for us to being with. But we needed more. We needed to extend it

Extending Customizer

Add in video (20 seconds)
(Widgets / customizer / contextual settings)

- Working closely with the XWP team at the start of our project we added a lot of functionality into customizer (some of which XWP have moved into WP core).
- This is a short video showing how our admins use customizer to build out a new page within WordPress.
- Play video (as we play)
    - All our sites utilize multiple content areas within our templates, yes pretty standard stuff.
    - Each content area can have a set of default widgets, which will be rendered on every URL of the website
    - We then have a plugin that allows WordPress admins to localise or override the widgets that show in a specific content area on a specific page/route. For example you can see the editor going to the /sports/ route, localising the widgets and having a completely different page structure for sports pages. We call these overrides, ‘contextual settings’ and are stored as a json_data structure in a custom post type in WordPress
    - Because we have ALOT of content on our sites (a little too much to be honest), we started to hit the memory limits. By default WordPress stores all of the widgets in the sidebars within the wordpress options table. This was a problem with us, as our entire sidebar was more than 1mb. 1mb is the default memory limit for each memcache key, this meant our wordpress_options was not being cached and causing huge

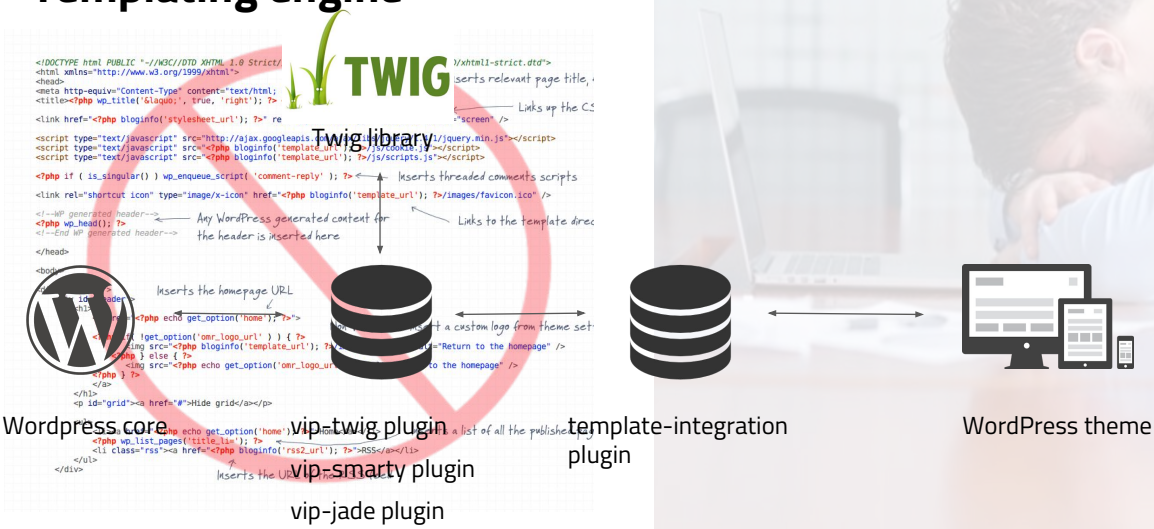
    - performance issues. We had to build another plugin that saves widget data in a new post_type called widget_instance so we now no longer need to store everything in wordpress options.
- Within our previous CMS it took hours to build a new page, within WordPress we can build a much more complex page within minutes

2.
Front-end architecture

Templating languages

- By the end of our first few months we had a solution for site build.
- Our front-end architecture was now key, we needed to ensure to the best of our ability we didn’t end up with a mess of front-end spaghetti code once our sites were rolled out.

Twig
Templating engine



- Make images more clear
- Challenge:
  - Based on all of our past experiences as developers here, I'm sure we can agree that using PHP as a templating language gets messy, really fast!
  - So we needed a templating engine to ensure clean decoupled front-end code.
- Solution:
  - We selected TWIG as our templating engine of choice. There are 2 plugins that we developed to power our twig setup.
    - Our first plugin called 'vip-twig' was responsible for integrating TWIG into WordPress.
    - Our second plugin 'template-integration' was responsible for exposing all of the wordpress core functions that a TWIG template was allowed to access (it basically bridged the gap between WordPress and the data we need to expose to the templates). If in the future need to support another templating language we can keep 'template-integration' plugin and build a new vip-smarty, or vip-jade plugin.
    - Using TWIG has really allowed us to decouple our front-end code from PHP, and it's an approach we recommend for any site using wordpress.

3.
Content is king

Keep the new site with relevant content.

- So we had an architecture that allowed our admins to build websites quickly within Customizer, we had decided on and implemented our templating engine...
- Now we need content!
- We had been working on our content ingestion functionality for around 6 months in parallel, and in May 2015 we were moving into beta testing in a VIP production environment, planning to launch our first site in the next month in June last year.
- As you can probably assume, based on the fact that our timeline up the top is no-where near the far right of the screen… yes we ran into some problems..

Content import challenges

- Display published stories on Wordpress in less than 60 seconds E2E
- Import process was duplicating Wordpress posts
- Initial content import had around 300,000 stories per site
- Needed to support up to 18 import updates per second per site

- In our first attempt, we had 2 major problems:
  - We had to display a news story on the WordPress front-end website in less than 60 seconds.
  - Secondly we were also getting duplicate content/posts being imported.
- Creating a new site and filling it up with content was not easy either as we need to import an initial dataset of around 300,000 posts from our old CMS and importing all this without SQL access was tricky.
- After this initial import we were going to have a constant stream of news story updates flowing through (around 3 updates per second), so we needed an architecture to support all this.

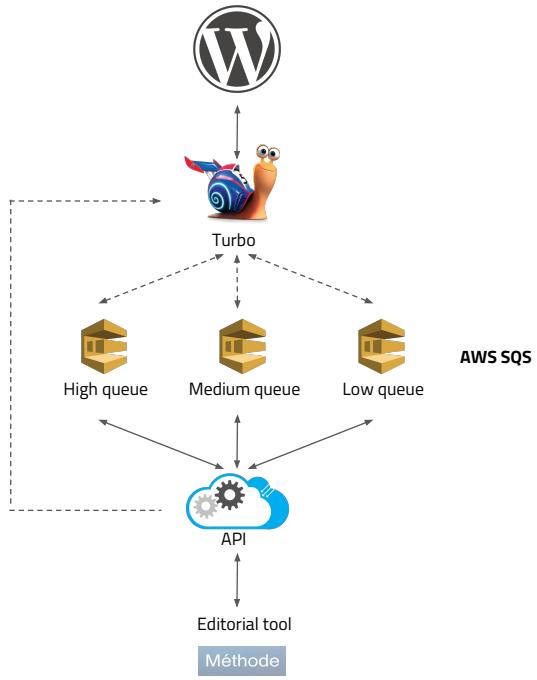
Pushing content to Wordpress

Turbo will receive that message, get the content from the API and push it into wordpress

Our APIs will push a message into SQS on update or delete

When saved it is sent to our APIs

Post gets created within our editorial tool



- So how have we solved all of these challenges:
  - Here is a simplified diagram of our architecture and it can be read from bottom to top (of course wordpress on top). This entire process of importing a post into WordPress is what we call end-to-end publishing.
  - (Click) First a journalist will create or update a story in our editorial tool. One thing that you need to understand here is that we don't always create content within wordpress (not just yet working on it). (Click)
  - Then that post will be sent to our APIs which they will organize the data and make it available to be consumed by other systems (WordPress being one of them). (Click)
  - Our APIs will then push notifications to SQS (Amazon queueing service) regarding the update.(Click)
  - On top of that we have developed a system with help of Wordpress VIP that is always listening to this queues and as soon as a message comes in it gets processed and imported directly into wordpress. It's basically a multi-threaded daemon.
- With this architecture in place, we are now able to:
  - have a single entry point in for all content imports, whether we are bulk importing 300,000 posts for a new site migration, or getting nearly 18 updates a second because a new story has just broke. It all runs through the same asynchronous process that we have solid unit tests

- around, real-time monitoring and trust that it is working. As soon as any queue gets more than 30 items in the queue our team starts getting notification alerts.
- We have the ability to push content into the high queue if its a breaking new story, or the low queue if its a bulk import.
- SQS will only give us a unique message so we have stopped in some way, the duplicate content issue we were facing.
- We can now ingest content end-to-end within less than 60 seconds, through all of our 4-5 platforms. We actually get most stories importing in under 10 seconds.

Content Schema

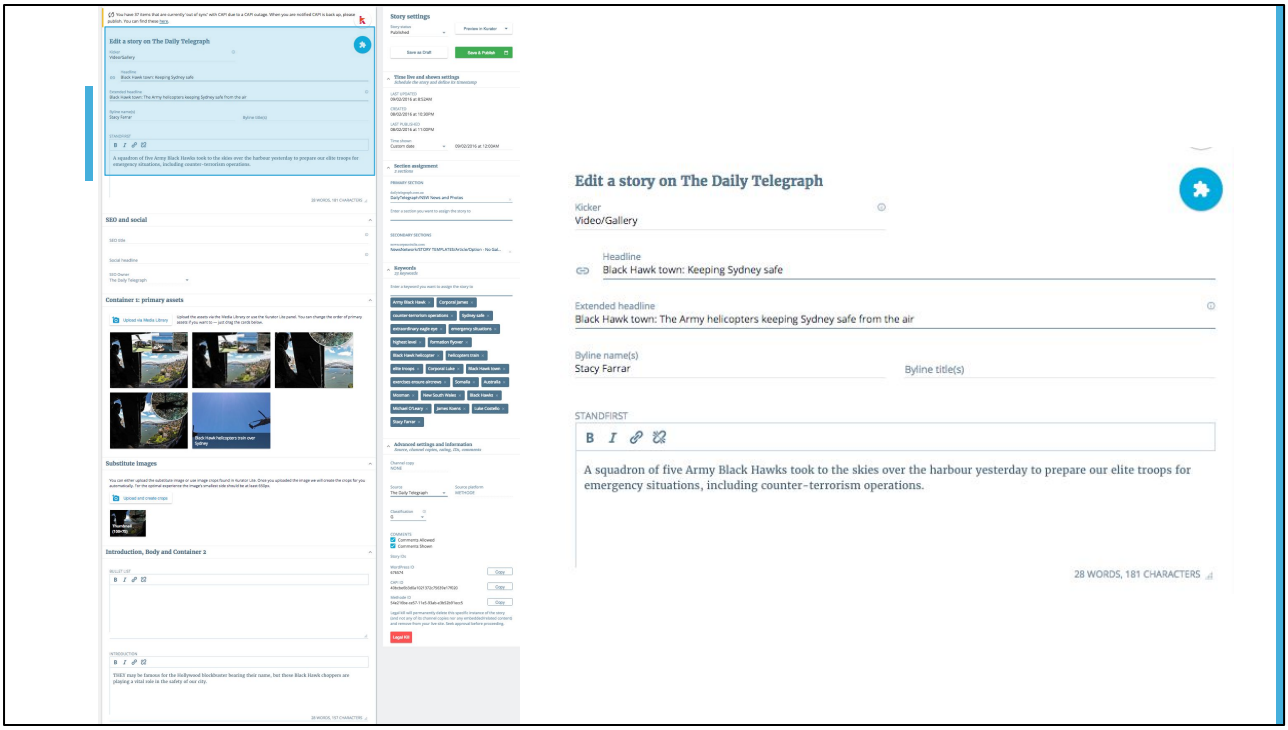
One object to rule them all

- Defined by our API's
- API's data is the source of truth
- Stored in post_content field (Revisions)
- One database call with all the fields instead of using post meta.


```
contentType: "NEWS_STORY",
+ id: {…},
originId: "1227356516274",
origin: "FATWIRE",
title: "mee apr27 story4 - EDITED IN FATWIRE",
subtitle: "extended - mee apr27 story4 - EDITED IN FATWIRE",
description: "There is nothing about Zootopia in need of fixing
illustrated thoughout.",
link: "http://www.newssit.com.au/entertainment/books/extended-me
paidStatus: "NON_PREMIUM",
originalSource: "ABC",
creditedSource: "news.com.au",
version: "DRAFT",
dateUpdated: "2016-06-02T21:22:00.000+00:00",
dateLive: "2016-04-26T21:19:00.000+00:00",
customDate: "2016-04-26T21:19:00.000+00:00",
dateCreated: "2016-04-26T21:25:00.000+00:00",
status: "ACTIVE",
+ thumbnailImage: {…},
related: [ ],
+ domainLinks: […],
+ primaryCategory: {…},
+ categories: […],
+ keywords: […],
+ authors: […],
+ domains: […],
+ references: […],
locationGeoPoints: [ ],
urlTitle: "extended--mee-apr27-story4",
revision: 2,
body: "<p>The city of Zootopia is inhabited by species of all si
animals of all species have learned to live as one.</p><p>In a l
at jobs that perfectly fit their place in nature.</p><p>Remarkbl
still a factor in this new world, however, which is why Zootopia
confined to merely writing parking tickets, Judy rapidly rises t
boss, Police Chief Bogo (Idris Elba).</p><p>With the equally rel
a vast conspiracy that threatens to divide Zootopian society ac
standFirst: "Disney's Zootopia stands out with its dazzling crea
kicker: "EDITED IN FATWIRE then add to collection",
commentsAllowed: false,
commentsTotal: 0,
commentsShown: false,
authorProfileIds: [ ],
bylineNames: [ ],
```


- Another challenge that we had to work with was the content schema of our posts. We had a complex data structure already provided by our APIs. First we thought on separating all the schema info into different post meta elements.
- But then we thought if we already have a json object the way we need it and every platform in our business understands it, why don't we use it, why reinvent the wheel, let's keep using that structure and keep our API's as our source of truth.
- (Click) You can see in our background image a trimmed down version of the json payload we have for each new story.
- This json object is stored in wordpress specifically in the post_content field in the database.
- This meant we needed to extend our wordpress setup to treat the post_content field as an object, and not as a simple text field. This meant changes within our plugins and our theme templates.
- Saving it there gave us the ability to have revisions of the whole object. And also we can do just one call to the database to get the whole object with all its meta.
- One thing that we kept as a rule was to use post_meta for searchable elements.
- All this gave us huge performance benefits, by more than halving the amount of database calls.

- As you can probably guess June past.. And we didn't launch our first site, but it stayed in beta testing. Our internal development team along with XWP and WordPress VIP team worked day and night for nearly 2-3 months to implement all the solutions we just spoke about in regards to content ingestion. And by September on one late, we launched our first site on WordPress VIP. IT was a night filled with anxiety, way too much pizza, and then a sigh of relief and satisfaction when everything switched over and it just worked..
- So we had successfully implemented a simplistic site built process, our front-end was still clean, and we had content quickly flowing into it.
- (click) We now need to use wordpress as it is meant to be, for editing content.
- Challenge:
    - As we spoke about, we had an existing complex data structure for each news story we had to maintain. We needed this information to be viewable and editable within Wordpres in the most simplistic form.



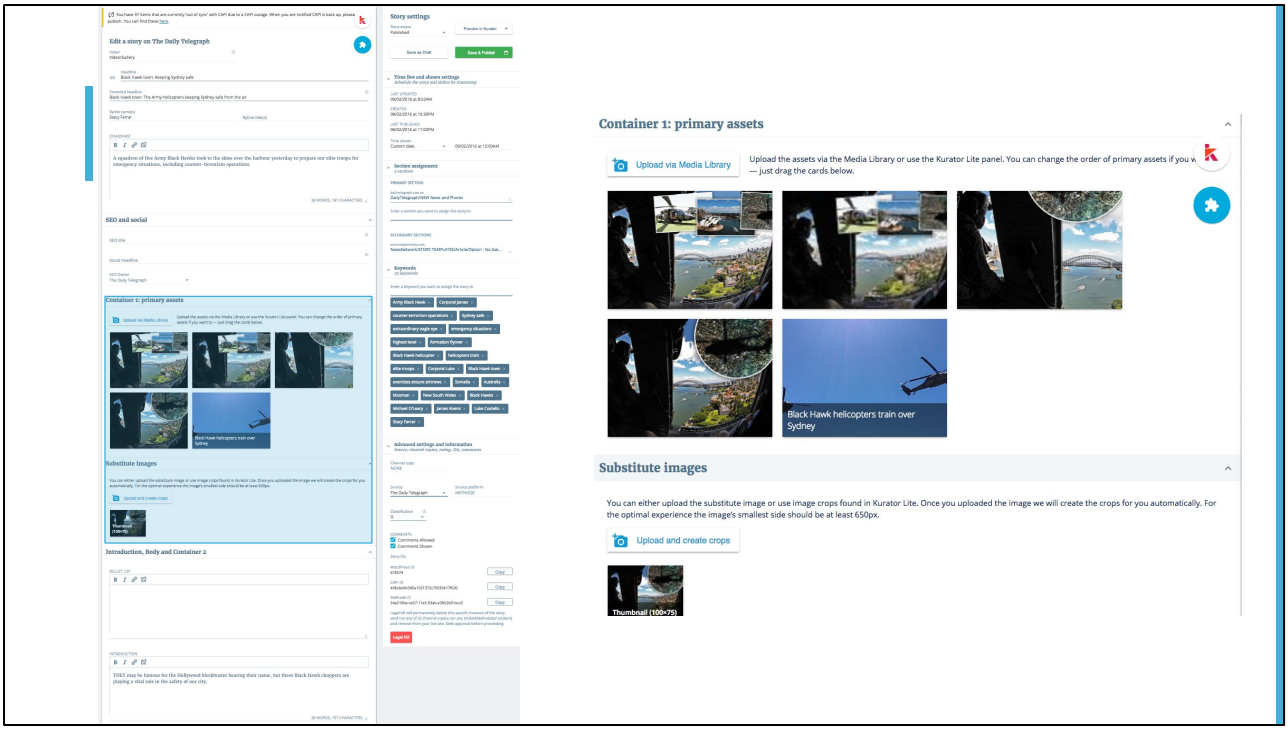
#Need to add old screens of authoring those look horrible

We developed an editor screen within wordpress admin for our JSON object, remember that we are storing this in post_content field .

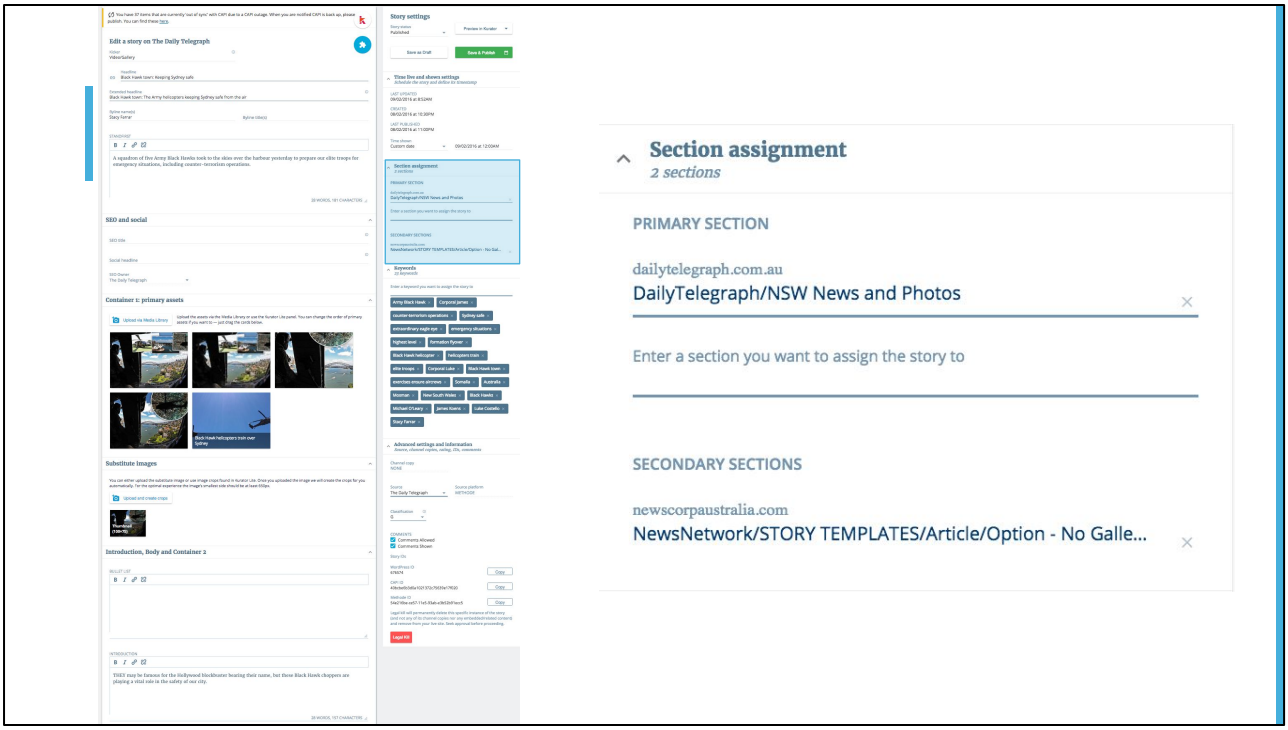
We originally built this in standard wordpress design until our designers checked out the new Calypso design. Then everything changed (meme image).

Our editing screen is very long sorry for the small image but the blue section is zoomed in the right side.

In this screen you'll be able to see a massive difference on how it looks, here you can see different post titles.

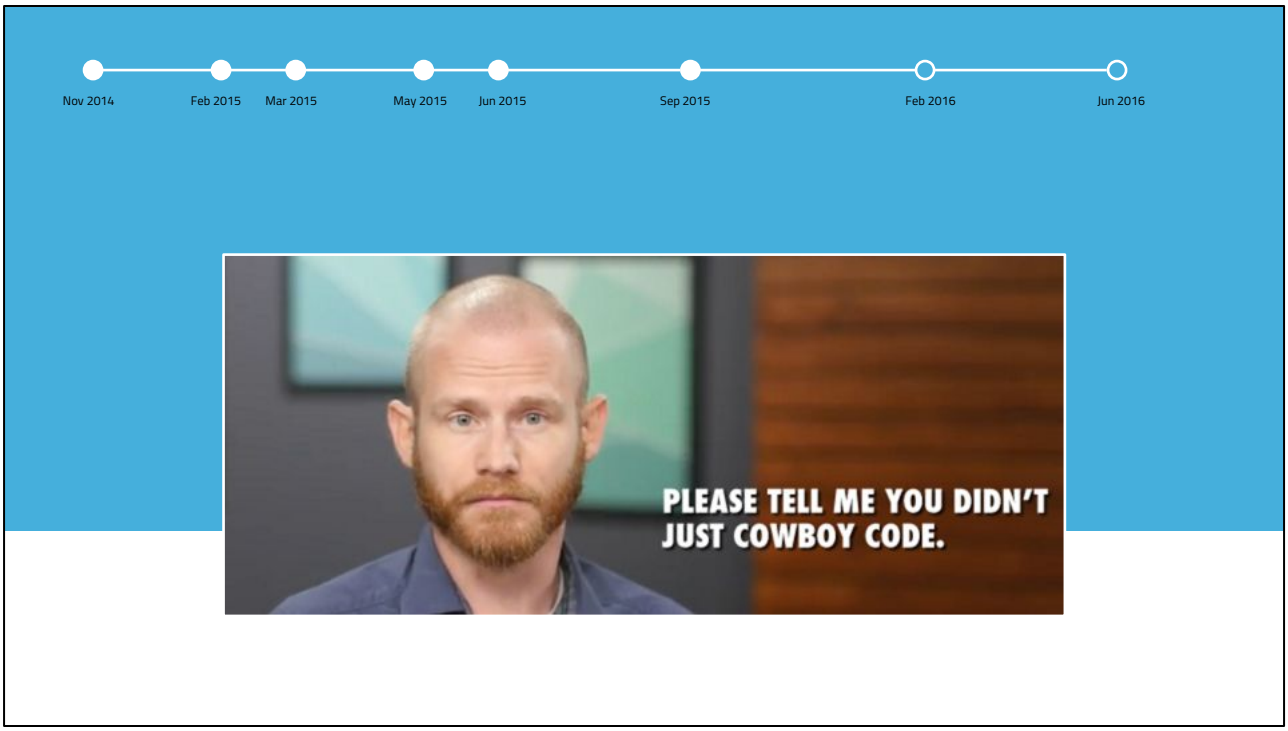


Also we manage multiple images which can be cropped and resized. This are features that wordpress already have but we improved on them.



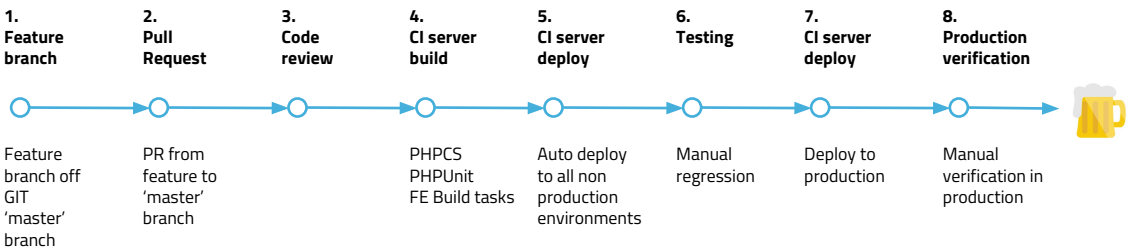
Our category management is different, of course we use wordpress categories but we also need to have a primary category.
This is for building the permalink of the post.

Our editors and publishers are loving this screen and they are being tested now.



- Within the first week of February 2016 we launched our last 4 sites on WordPress VIP in 1 week - everything went smoothly.
- Then gradually over the next 4 months until last week we developed and rolled out all of the authoring functionality in wordpress admin that we just showed you.
- So we needed maintain this WordPress monster we have created
- (click) we also needed to at all costs make sure it was impossible for any cowboy coders to rise up and take control.
- (click) For this, it was critical to have a solid continuous integration and deployment setup.
- We need to be able to get code from a developers macbook all the way to production in the least risky and most stable way.
- At it's core it comes down to feature branching and feature based deployments. Each developer owns their feature from development, through testing all the way to releasing it into production. We never throw a feature over the fence to a tester or a release manager to take over - this reduces ownership of the feature

Deploying stable features often



- Every feature we build always sits in it's own feature branch (branched off master), it is always tested in isolation and can be deployed at any time in isolation. We never bundle up multiple features in a single release, as this adds risk and too many dependencies.
- Our team utilizes Pull Requests to control any code merges into master branch, and this is where every line of code is code reviewed by another developer in the team. No line of code reaches production without being first code reviewed.
- We use the Bamboo continuous integration and deployment platform by Atlassian to run all of our automation.
- On every code check-in, we run PHPCS to ensure coding standards are followed, we run PHPUnit to ensure all unit tests are passing and there are no regression issues. Once all our automated tests pass within Bamboo we package up the codebase (running front-end build tasks, compiling our twig templates), and these changes are auto deployed to all of our 50+ non-production servers.
- Once any manual regression testing passes, we again use bamboo to automatically tag the release for this feature and push it through our deploy pipeline into the WordPress VIP production environment.
- The best part? This wasn't the process we started with 18 months back. This evolved form countless retrospectives, team huddles, issues we had with our current process to what it is now, and it works pretty well.

Welcome to...
The Feature Toggle

Define a new feature

```php
use NewsCorpAU\Features_Manager\Features_Manager;

// Define a new feature
Features_Manager::set_feature_toggle(
    'authoring.auto-sharpen-images', // Feature name
    'Enable this to automatically sharpen images', // Feature description
    false // Default active state
);
```

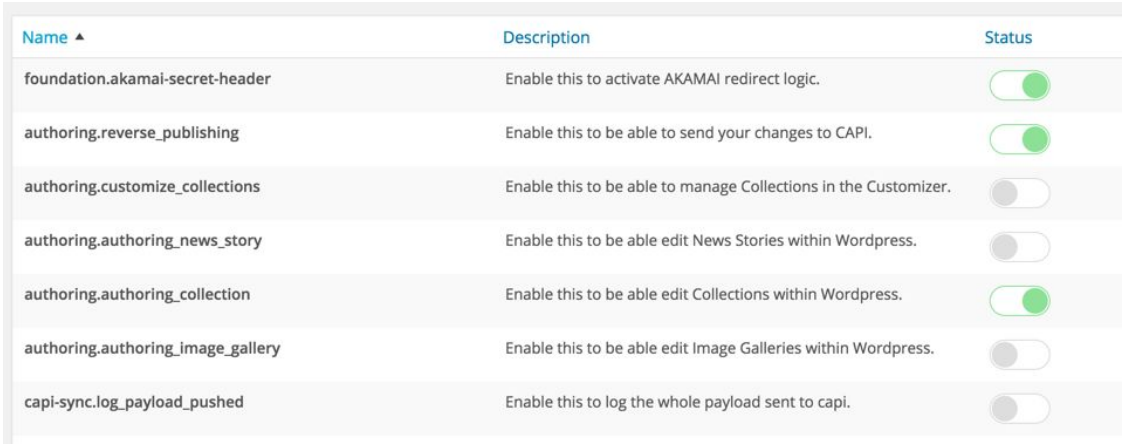
Check if feature is activated

```php

if ( Features_Manager::is_active( 'authoring.auto-sharpen-images' ) ) {
    // Now we can assume this feature is active
}
```

- 1 minute
- Problem/Challenge:
    - Within WordPress VIP, you cannot control the exact time your code is deployed to production. If your feature has changes in 4 different repositories you also cannot always guarantee the order it will be deployed in. Besides creating backwards compatible code, we needed a way to toggle new features on and off on certain production sites.
- Solution:
    - Welcome to the feature toggle.
    - It was developed by Roman, one of the amazing developers in our team back in Australia, and is an implementation of the FeatureToggle technique from Martin Fowler
        - See http://martinfowler.com/bliki/FeatureToggle.html for more details
    - This plugin abstracts all the complexities of feature toggling so you can implement it with just a few lines of code.
    - Image one: Within just a few lines of code, your plugins/theme can define a new feature which is stored within the wordpress options, and it can be deactivated by default.
    - Image two: Your plugins and theme code now has access to check if the specific feature is enabled as you can see above

Welcome to...
The Feature Toggle



Name ▲	Description	Status
foundation.akamai-secret-header	Enable this to activate AKAMAI redirect logic.	
authoring.reverse_publishing	Enable this to be able to send your changes to CAPI.	
authoring.customize_collections	Enable this to be able to manage Collections in the Customizer.	
authoring.authoring_news_story	Enable this to be able edit News Stories within Wordpress.	
authoring.authoring_collection	Enable this to be able edit Collections within Wordpress.	
authoring.authoring_image_gallery	Enable this to be able edit Image Galleries within Wordpress.	
capi-sync.log_payload_pushed	Enable this to log the whole payload sent to capi.	

- 30 seconds
- Better images
- Finally the plugin exposes a Wordpress admin screen that lists all of the features currently available along with their activated/de-activated state.
- Using this feature toggle plugin allows us to gradually roll out large features to all of our sites, staggering our regression testing each site at a time.

In summary

1. Migrations are hard hard work...
2. WordPress can scale for the enterprise
3. Lastly, have great partners

- Migrations are hard hard work…
  - They take a lot of planning and you will always run into unknowns and unknown unknowns, no matter how much you plan it.
  - Getting code to production ASAP allowed us to validate your solutions in an end-to-end beta/production environment.
  - You need a strong, high performing engineering team of developers, BAs, testers and PMs, who genuinely love what they do.
- People says wordpress isn't scalable, but we, at least in our use case have proven that it does - it scales well if you decide on the correct architecture.
- For us we partnered with 2 strong partners, XWP, an amazing team of developers who know Wordpress really well.
  - They were fundamental in getting our sites initially setup.
    - NewsCorp and XWP worked as a single team, we were so tightly integrated in regards to communications, scoping, development and testing. Don't treat teams outside your office as separate teams.
  - Secondly, the WordPress VIP team was another strong allie. You need to work well to establish a strong relationship with them. We won't lie, we had some tough times in the beginning
    - When we struggled working with them mimicing the production wordpress.com infrastructure in our non production

    - environments.
    - And them with us, when we would be send them 100’s of 1000’ s of lines of code to review, all day every day.
    - But they stood strong and we stood strong and over the course of 6-9 months we have forged a great relationship.
- With both the XWP and WordPress VIP teams we can collaborate together to solve any problem or challenge we face if needed..

Thanks!!

Any questions?

Find us on Twitter

@shoeazap & @dionbeetson